University of Macau

Faculty of Science and Technology



Robust Outdoor Smart Parking Lots Detection System Under Illuminance VarianceRobust Outdoor Smart Parking Lots Detection System Under Illuminance Variance

by

TAI CHON KITTAI CHON KIT, Student No: DB625948DB625948

Final Project Report submitted in partial fulfillment  
of the requirements of the Degree of   
Bachelor of Science in Computer Science

Project Supervisor

Prof. Chi Man VONG

05 June 2020

# 

DECLARATION

I sincerely declare that:

1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name : TAI CHON KIT

Student No. : DB625948

Date : 05 June 2020

ACKNOWLEDGEMENTS

The author would like to express his utmost gratitude to UM for providing the opportunity to carry out a project as a partial fulfillment of the requirement for the degree of Bachelor of Science.

Throughout this project, the author was very fortunate to receive the guidance and encouragement from his supervisor Prof. Chi Man VONG.

**ABSTRACT**

This report proposes a neural network using convolution algorithm as a parking lot occupancy detection method and trains the classification model using the "CNR-EXT" data set. Experiments show that the improved model can not only classify and detect the occupancy of parking Spaces, but also effectively classify and detect other data sets such as "PKLot". At the same time, although the model USES private parking Spaces as training, it can effectively judge the parking situation of motorcycle parking Spaces.

TABLE OF CONTENTS

CHAPTER 1. INTROUDUCTION 9

CHAPTER 2. RELATED WORK 10

2.1 Deep Learning 10

2.2 Convolutional Neural Network 10

2.3 Regularization 10

2.4 Dropout 10

2.5 BatchNormalization 10

2.6 Model car dataset 11

CHAPTER 3. Project Execution Schedule 12

CHAPTER 4. Function Specification 13

4.1 Get pictures from the camera 13

4.2 Cropping the grid from the picture 13

4.3 Parking space status check 13

4.4 Display results 13

CHAPTER 5. Software Design Specification 14

5.1 Overview 14

5.2 Building Past Classification Model 14

5.3 Building Improve classification model 15

CHAPTER 6. Implementation Narrative and Description 17

6.1 Overview 17

6.2 Crop the image 17

6.3 Past Classification Model 19

6.4 Improve Classification Model 20

6.5 Displaying Result 21

CHAPTER 7. System Quality 24

7.1 Past Classification model 24

7.2 Improve Classification model 25

7.3 Compare accuracy of two models 26

7.4 Check for irregular parking 27

CHAPTER 8. Ethics and Professional Conduct 28

CHAPTER 9. Summary 29

CHAPTER 10. References 30

LIST OF FIGURES

[Figure 1: Example of Image Cropping with dataset 13](#_Toc41843094)

[Figure 2: Example of Parking space status check 13](#_Toc41843095)

[Figure 3: Example of parking lot detcet result 13](#_Toc41843096)

[Figure 4: Sample input image for cropping image with moder car 18](#_Toc41843097)

[Figure 5: Sample Detection Result with model car 23](#_Toc41843098)

[Figure 6: Past Classification Model training and validation accuracy 24](#_Toc41843099)

[Figure 7: Past Classification Model training and validation loss 25](#_Toc41843100)

[Figure 8: Improve Classification Model training and validation accuracy 26](#_Toc41843101)

[Figure 9: Improve Classification Model training and validation loss 26](#_Toc41843102)

[Figure 10: irregular parking sample 27](#_Toc41843103)

LIST OF TABLES

[Table 1: Summary of Dataset 14](#_Toc41843104)

[Table 2: Output Shade and Parameters of Layer in Convolution Neural Network 15](#_Toc41843105)

[Table 3Output Shade and Parameters of Layer in Convolution Neural Network 16](#_Toc41843106)

[Table 4: Sample Data of csv file with model car 18](#_Toc41843107)

[Table 5: Summary of experiment result (Past Classification Model) 25](#_Toc41843108)

[Table 6: Summary of experiment result (Improve Classification Model) 26](#_Toc41843109)

[Table 7: Compare Accuracy of two models 27](#_Toc41843110)

# INTROUDUCTION

As technology continues to advance, people's living standards continue to rise. As the Macau government continues to promote smart city plans. One of the key points is smart parking that is one of the indispensable components in smart city plan applications, and the core technology on that is too smart parking lots detection system to detect the occupation in the parking lot.

Especially, a suitable intelligent parking lot detection system is important for Macau. Base on the data in Statistics and Census Service of Macau[1], the number of motor vehicles in Macau is more than 240 thousand, including near 117 thousand automobiles and 124 thousand motorcycles. Moreover, the Macau population only has 696 thousand. This is a scary data. It means that closely every three Macau people own one vehicle on average. Those data indicated the parking space must be insufficient. It highlighted an important issue of a parking lot detection system.

Though there are many approaches to determine the occupancy status of parking space, such as equipped with sensors in every parking lot. We consider that using ground sensors (for example, magnetic sensors) in each parking space is a good method for accurately detecting parking lot occupancy. However, this method will require high costs, human resources, and time due to the installation of the sensor in each parking space and continuous maintenance.

In contrast, smart cameras can monitor multiple parking spaces simultaneously. And the cost is much lower than the cost of installing and maintaining sensors in the whole parking lot. Because smart cameras are very common in society nowadays. Especially in Macau, Macau Government and Macau Police Station jointly installed more 1600 monitor that called ‘sky eye’ in recent years. It can cover the whole Macau more effectively. Therefore, we believe that the use of smart camera detection will be a great viable option.

Vehicle detection in the parking lot is to identify the vehicle in the image obtained by a smart camera. However, it is a very difficult work that located multiple vehicles of different sizes in the foreground and background of an image. Also, the detection system with Convolutional Neural Network (CNN) is very suitable for overcoming this issue. CNN has been proven that can learn important features in images effectively. The advantage of implement deep CNN directly on smart cameras is reducing the amount of data transferred.

In the data, we used a dataset called CNRPark-EXT to train a CNN classification model.[2] This dataset was taken by 9 smart cameras. These cameras have several perspectives, different weather and time, some occlusion and shadows make occupancy detection tasks more challenging under various conditions (such as illuminance variance in different dates).

To evaluate the classification model, we used the datasets called CNR-EXT and PKLot. PKLot includes approximately 695,900 parking space images. [3Those images come from the two Italy parking lot. Also, it is similar to the CNR-EXT dataset that includes different perspectives, brightness, weather, occlusion, and shadows. Implementation pointed out the classification model works well in both datasets.

# RELATED WORK

## Deep Learning

Deep learning is a branch of machine learning. It is an algorithm that uses artificial neural networks as an architecture to perform representation learning on data.

Deep learning is an algorithm in machine learning based on the representation learning of data. Observations can be expressed in a variety of ways, such as a vector of intensity values ​​for each pixel, or more abstractly as a series of edges, regions of a specific shape, etc. It is easier to learn tasks from examples using some specific representation methods.

## Convolutional Neural Network

Convolution Neural Network (CNN) can complete the computer vision task effectively. Many image recognition models are also based on CNN's architecture to extend. It is also worth mentioning that the CNN model is also a few deep learning models built with reference to the human visual organization

## Regularization

When there is not enough training data, or overtraining, it often leads to overfitting. As the training process progresses, the complexity of the model increases, and the error on the training data gradually decreases, but the error on the validation set gradually increases. Because the trained network overfits the training set, it does not work on the data outside the training set. Regularization is to solve the problem of overfitting. [6] Using the regularization method will automatically weaken unimportant feature variables, automatically "extract" important feature variables from many feature variables, and reduce the magnitude of feature variables. This method is very effective. When we have many characteristic variables, each of them can have a little influence on the prediction. As seen in the example of housing price prediction, we can have many characteristic variables, each of which is useful, so we do not want to delete them, which leads to the concept of regularization.

## Dropout

Dropout is a regularization for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is an efficient way of performing model averaging with neural networks. [7] During each training, half of the feature detectors are stopped Work, so that the emergence of one neuron should not depend on another neuron, improve the generalization ability of the network.

## BatchNormalization

First, the Batch Normalization algorithm normalizes each layer of input in each iteration, normalizing the distribution of input data to a distribution with an average value of 0 and a variance of 1, as follows:

A close up of a logo

Description automatically generated

Among them, represents the k-dimension of the input data, represents the average value of the dimension, Indicates the standard deviation.

However, this approach has a fatal disadvantage. Although the data distribution of each layer is fixed in this way, this distribution is not necessarily the data distribution of the previous layer to be learned, so that forced normalization will destroy the just The learned feature, the second step of the Batch Normalization algorithm solves this shortcoming.

The Batch Normalization algorithm sets the sum of two learnable variables and in the second step, and then uses these two learnable variables to restore the data distribution that should be learned in the previous layer.[8]

A picture containing object, clock

Description automatically generated

The purpose of adding this operation is to restore the data distribution that needs to be learned in the previous layer, so that Batch Normalization converts all the original non-fixed data distribution into a fixed data distribution, and this data distribution is exactly the distribution to be learned.

## Model car dataset

Paper and model cars are used to simulate the environment of outdoor parking lots. The types of model cars are private cars, motorcycles, taxis, buses and trucks. These model cars are common on the streets of Macau. Then set up the tripod and camera, shoot the simulated outdoor parking lot, make a simple data set by yourself, and then use the system to plant and cut the photos, and use the trained model to detect the parking space occupation.

# Project Execution Schedule

Task.0: teach yourself Python, Numpy for programming skills, and Keras for deep learning frameworks (please learn the attached book)

Learn about Python and its library, Numpy, from Justin Johnson's online tutorial at the university of Michigan, and understand a book called "Python deep learning."

Task 1: implement the classification model based on CNN

The Python library (called Kera) used to build the deep learning network is applied to create a basic binary classification model for occupancy detection (only the occupied state is classified as "busy" or "idle").

Task 2: collect data sets for outdoor parking lots with lighting variations

According to online resources, there are two image sets, namely the cnr-ext dataset and the PKLot dataset.

Task.3 \* : improve the classification model

The generalization of the model is improved by introducing batch normalization layer and zero filling layer, which makes the network more robust to parameters and activation functions

Task.4 \* : system evaluation

Try to compare the accuracy of the classification model under different parameters and data sets. Therefore, the classification model and several conditions are adjusted to improve the accuracy between the test data and the classification model.

\* personal contribution.

# Function Specification

## Get pictures from the camera

Through the camera lens of the outdoor parking lot, you can get a picture of the parking lot. The system uses the picture to determine the status of the parking lot, so the picture is used for later detection.

## Cropping the grid from the picture

Because the camera will capture a lot of parking spaces at once, it is necessary to cut out each parking space and inspect them one by one.

A screenshot of a cell phone

Description automatically generated

Figure 1: Example of Image Cropping with dataset

## Parking space status check

The pictures of the parked parking spaces are sent to the trained classification model for detection. In the detection, the input is a parking space image and the probability of being a busy parking lot. Because it is in an outdoor parking lot, the system can be correctly detected in different weather and time.

A picture containing food

Description automatically generated

Figure 2: Example of Parking space status check

## Display results

According to the output of the model, the pictures in the parking lot show the detection of each parking space. For example, green stands for free, and red stands for busy.

A picture containing sitting, black, white

Description automatically generated

Figure 3: Example of parking lot detcet result

# Software Design Specification

## Overview

In order to detect the status of each parking space in the image of the parking lot, first, cut the parking lot in the image into 150x150 pieces. The patch is then sent to the classification model to obtain the occupancy probability, which determines whether the patch is busy or idle. Finally, the occupancy status of each parking space is displayed on the input image, with the red box representing busy and the green box representing free.

Initially, the classification model was trained using a dataset called CNR-ext. The data set doesn't have many images of strong sunlight. In order to improve the accuracy of sunny days, the normalization method of BatchNormalization layer is added, which can effectively reduce the over-fitting of models.

CNR-EXT is a data set of parking spaces.[2] The database has a total of 144,965 images. The image is taken from a parking lot in Italy. CNR-EXT consists of images collected by 9 cameras with different angles of view and different angles of view from November 2015 to February 2016 under different weather conditions. CNR-EXT captures different conditions of light conditions and includes partial occlusion modes due to obstacles (trees, lampposts, other cars) and partially or globally shadowed cars.

PKLot is also a database of parking Spaces. The database captured images of sunny, cloudy and rainy days in two different parking lots in Brazil. A total of 12,417 images (1280X720) were captured. By splitting parking Spaces using an XML file, you get 695,899 images of parking Spaces. These photos come from two parking lots in Brazil. Like the CNR-EXT data et, these images also contain images taken under different weather and brightness conditions.

Since the two databases are from different regions, CNREXT is used to train the model, and PKLot can be used as a generalization test for the model. Then PKLok is divided into the bright dataset and dim dataset according to its brightness to detect Robust of the model.

Table 1: Summary of Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | CNR-EXT | PKLot | PKLot(Bright) | PKLot(Dim) |
| Busy Space | 79,307 | 337,780 | 60,296 | 164,534 |
| Free Space | 65,658 | 358,119 | 119,570 | 118,456 |
| Total | 144,965 | 695,899 | 179866 | 282990 |

## Building Past Classification Model

The classification model is used to determine the occupancy status of parking spaces. It consists of several layers, including the convolution layer, maximum pooling layer, batch normalization layer, Zero Padding layer, fatten layer, dropout layer, and dense layer.

This is the architecture of the past classification model in this project. The input is a parking lot image, and the output is a busy probability.

Table 2: Output Shade and Parameters of Layer in Convolution Neural Network

|  |  |  |
| --- | --- | --- |
| Layer(type) | Output Shape | Parm # |
| Conv2d\_1(Conv2D) | (None, 148, 148, 32) | 896 |
| Max\_pooling2d\_1(MaxPooling2) | (None, 74, 74, 32) | 0 |
| Conv2d\_2(Conv2D) | (None, 72, 72, 64) | 18496 |
| Max\_pooling2d\_2(MaxPooling2) | (None, 34, 34, 128) | 0 |
| Conv2d\_3(Conv2D) | (None, 17, 17, 128) | 73856 |
| Max\_pooling2d\_3(MaxPooling2) | (None, 15, 15, 128) | 0 |
| Conv2d\_4(Conv2D) | (None, 7, 7, 128) | 147584 |
| Max\_pooling2d\_4(MaxPooling2) | (None, 6272) | 0 |
| Flatten\_1(Flatten) | (None, 6272) | 0 |
| Dense\_1(Dense) | (None, 512) | 3211776 |
| Dropout\_1(Dropout) | (None, 512) | 0 |
| Dense\_2(Dense | (None, 1) | 513 |

The convolutional layer is a set of parallel feature maps, which are composed by sliding different convolution kernels on the input image and performing certain operations. Furthermore, at each sliding position, an element corresponding product and sum is performed between the convolution kernel and the input image to project the information in the receptive field to an element in the feature map. This sliding process can be called step Z\_s, which is a factor controlling the size of the output feature map. The size of the convolution kernel is much smaller than the input image, and overlapped or parallel ACTS on the input image. All elements in an feature map are calculated by a convolution kernel, that is, an feature map shares the same weight and bias terms.

The maximum pooling layer is to down-sample the input. The sliding window slides on the input to find the maximum value of each color patch, thereby reducing the size of the input as the output. MaxPooling is a method to reduce the redundancy of the model. Take 2x 2 MaxPooling as an example. If the figure is a 4x4 input matrix, the 4x4 matrix will be divided into 2x2 sub-matrices composed of two rows and two columns, and then each 2x2 sub-matrix takes a maximum value as a representative, thus obtaining a two Row and two column results:

Without a maximum pooling layer, it is difficult to learn the elements in the spatial hierarchy. In addition, the larger the parameters in the model, the more likely it is to lead to overfitting.

The Flatten layer is used to convert the input shadows of 3D tensors to 1D tensors because the input shadows of the tightly connected classifiers are 1D tensors.

The dropout layer is generally added to the fully connected layer to prevent overfitting and improve the model generalization ability.

Dense layer is a layer closely connected in a simple neural network. Also known as full connection layer (fully connected layer (FC)) in the whole convolutional neural network series "classifier" role. If operations such as convolution layer, pooling layer and activation function layer map the original data to the hidden layer feature space, the full connection layer will map the learned "distribution feature representation" to the role of sample marker space. In practice, the full connection layer alternate convolution operation is realized: for the full connection layer with full connection at the front layer, the convolution kernel is 1x1; While the front layer is the full connection layer of the convolution layer, which can be converted into volume convolution with the convolution kernel being HXW, h and W are the height and width of the convolution result of the front layer respectively.

It's essentially a linear transformation from one eigenspace to another eigenspace.

Each dimension of the target space - that is, a cell in the hidden layer - is thought to be affected by each dimension of the source space. Regardless of rigor, you could say that the target vector is the expected sum of the source vectors.

In CNN, full connection often appears in the last few layers, which is used to add and add the features of the previous design. For such MNIST, convolution and pooling in the front are equivalent to feature engineering, while full connection in the back is equivalent to feature reduction. In the binary classification problem, the model should end with a dense layer activated by 1 unit Sigmoid, and give the probability of the predicted result.

## Building Improve classification model

The improve classification model is used to determine the occupancy status of parking spaces. It consists of several layers, including the convolution layer, maximum pooling layer, batch normalization layer, Zero Padding layer, fatten layer, dropout layer, and dense layer.

This is the architecture of the improve classification model in this project. The input is a parking lot image, and the output is a busy probability.

Table 3Output Shade and Parameters of Layer in Convolution Neural Network

|  |  |  |
| --- | --- | --- |
| Layer(type) | Output Shape | Parm # |
| Conv2d\_1(Conv2D) | (None, 148, 148, 32) | 896 |
| batch\_normalization\_1(Batch\_Normalization) | (None, 148, 148, 32) | 128 |
| Max\_pooling2d\_1(MaxPooling2) | (None, 74, 74, 32) | 0 |
| Conv2d\_2(Conv2D) | (None, 72, 72, 64) | 18496 |
| batch\_normalization\_2(Batch\_Normalization) | (None, 72, 72, 64) | 257 |
| Max\_pooling2d\_2(MaxPooling2) | (None, 36, 36, 64) | 0 |
| Conv2d\_3(Conv2D) | (None, 34, 34, 128) | 73856 |
| batch\_normalization\_3(Batch\_Normalization) | (None, 34, 34, 128) | 512 |
| Max\_pooling2d\_3(MaxPooling2) | (None, 17, 17, 128) | 0 |
| Zero\_padding2d\_1(Zeropadding) | (None, 19, 19, 128) | 0 |
| Conv2d\_4(Conv2D) | (None, 17, 17, 128) | 147584 |
| batch\_normalization\_4(Batch\_Normalization) | (None, 17, 17, 128) | 512 |
| Max\_pooling2d\_4(MaxPooling2) | (None, 8, 8, 128) | 0 |
| Zero\_padding2d\_2(Zeropadding) | (None, 10, 10, 128) | 0 |
| Conv2d\_5(Conv2D) | (None, 8, 8, 256) | 295168 |
| batch\_normalization\_5(Batch\_Normalization) | (None, 8, 8, 256) | 1024 |
| Max\_pooling2d\_5(MaxPooling2) | (None, 4, 4, 256) | 0 |
| Conv2d\_6(Conv2D) | (None, 2, 2, 256) | 590080 |
| batch\_normalization\_6(Batch\_Normalization) | (None, 2, 2, 256) | 1024 |
| Max\_pooling2d\_6(MaxPooling2) | (None, 1, 1, 256) | 0 |
| Flatten\_1(Flatten) | (None, 256) | 0 |
| Dense\_1(Dense) | (None, 512) | 131584 |
| Dropout\_1(Dropout) | (None, 512) | 0 |
| Dense\_2(Dense | (None, 1) | 513 |

Zero Padding layer is used to maintain the size of the output graphics, and also has the effect of suppressing the edge of the image, because in general, we think that the middle part of the image is more important than the surrounding.

The batch normalization layer is used to adjust the distribution of the data, regardless of the activation function. It normalizes the output of each layer to a distribution with a mean of 0 and a variance of 1. This ensures the effectiveness of the gradient and reduces the initialization of the parameters. rely on. Training is faster and higher learning rates can be used. Increased generalization ability.

Batch Normalization normalizes the input of each layer of the network to ensure that the mean and variance of the input distribution are fixed within a certain range, reducing the problem of Internal Covariate Shift in the network, and alleviating the gradient disappearance to a certain extent. , Accelerates model convergence; and BN makes the network more robust to parameters and activation functions, reducing the complexity of neural network model training and parameter adjustment; finally, the use of mini-batch mean / variance as the overall The estimation of sample statistics introduces random noise, which has a regularized effect on the model to a certain extent.

# Implementation Narrative and Description

## Overview

Python is a major programming language used in outdoor parking systems to cut images, generate and train models, generate images to enhance data, and detect parking Spaces.

To process images, a Python library called Numpy is used.[9] NumPy is the base package for scientific calculations using Python. This library is important in our parking space system because all images need to be converted to the Numpy array to perform the calculus. After conversion, you can crop the image and use it as input to the model.

To build and train models, a Python library named TensorFlow was used and then introduced into the Keras Library[10], an advanced neural network API written in Python that enables rapid and easy model generation. In addition, it supports convolutional and cyclic networks and runs seamlessly in the CPU of a typical home computer. It can also be configured with a GPU to increase speed. In this model training, we mainly used GPU to train the model, and stored the data on SSD to improve the speed of related I/O.

To display system output, OpenCV Library is used in Python. [11] It is an open source library of computer vision and machine learning software. It supports a variety of programming languages, including Java, C ++, Python and MATLAB. In the report, we mainly used Python to draw a long box on the image to show the occupancy status of the parking space.

The IDE used to develop the system is Anaconda. [12] Therefore, it takes a lot of time to train the model. To reduce the time spent on training models, the system uses NIVIDA RTX 2070 super for model training.

## Crop the image

First, know the coordinates of each parking space in the image, so use CSV file for recording. The document should include the parking space number, the parking space in the upper left corner of the image x and y coordinates as the starting point, the parking space in the lower right corner of the image x and y coordinates as the end point, subtract the two values to get the length and width of the parking space. After recording the coordinates of the parking space, the program first reads the data from the CSV file and displays it line by line. The camera image is then converted to a Numpy array by Numpy. Finally, read the CSV file data, crop the image of the correct parking space in the image, and temporarily store it in the system location.

A picture containing indoor, sitting, table, white

Description automatically generated

Figure 4: Sample input image for cropping image with moder car

Table 4: Sample Data of csv file with model car

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SlotId | X | Y | W | H |
| 1 | 1700 | 1038 | 256 | 470 |
| 2 | 2011 | 1015 | 289 | 455 |
| 3 | 2365 | 982 | 279 | 456 |
| 4 | 2676 | 950 | 316 | 451 |
| 5 | 3012 | 903 | 312 | 456 |
| 6 | 1663 | 2298 | 310 | 651 |
| 7 | 2044 | 2280 | 310 | 637 |
| 8 | 2430 | 2228 | 330 | 637 |
| 9 | 2855 | 2187 | 337 | 637 |
| 10 | 3266 | 2159 | 362 | 623 |

How does this run the code for the implementation method：

import numpy as np

import cv2

import sys

import csv

import os

import time

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

from PIL import Image

from IPython.display import clear\_output

from tensorflow.keras import models

# Saving output path

save\_dir = './test2'

path\_source = './modelcar/'

path\_location = 'modelcar\_cam.csv'

start = time.time()

count = 0

patch\_size = (150, 150)

# size in resizing patch

# original image is 1000x2592 in size

font = cv2.FONT\_HERSHEY\_COMPLEX # for better font display

# construct "bbox\_set2" dict

bbox\_set2={}

with open(path\_location) as csvfile:

reader = csv.reader(csvfile)

for idx, line in enumerate(reader):

if idx==0:

continue

# add into bbox\_set

bbox\_set2[int(line[0])]= [int(line[1]), int(line[2]), int(line[3]), int(line[4])]

# Perform classification for each parking lot

for files in os.listdir(path\_source):

if files.endswith('.JPG'):

image = path\_source + files

# Perform classification for each parking lot

img = cv2.imread(image)

patches = {} # create a dictory to store all the patches

# extract patches

for slotId, bbox in bbox\_set2.items():

bbox\_set3 = (np.asarray(bbox)).astype(int

patch = img[bbox\_set3[1]:bbox\_set3[1]+bbox\_ set3[3],  
bbox\_ set3[0]:bbox\_ set3[0]+bbox\_ set3[2]]

patch = cv2.resize(patch, patch\_size)

# Attach the patch to the patch dictionary

patches[slotId] = patch

# Convert the image to RGB format so that it can be input into keras training compatibility

patch\_input\_rgb = cv2.cvtColor(patch, cv2.COLOR\_BGR2RGB)

patch\_input\_rgb = patch\_input\_rgb /255.0

# rescale to (0,1) range

count += 1

save\_d = p2 + '/output' + str(count) + '.jpg'

# save image

cv2.imwrite(save\_d, patch)

# Clear output when image shows more than 250

if count % 50 == 0:

clear\_output(wait=True)

#display patch

patch\_display = cv2.cvtColor(patch, cv2.COLOR\_BGR2RGB)

plt.imshow(patch\_display)

plt.show()

#cv2.imwrite(outputimg, img)

print(str(savd\_d) + ' is save! The time is: ' + str(time.time() - start))

## Past Classification Model

To train past classification models, CNR-EXT data sets were used. 94,493 images were extracted from the database as the training set of the model. The CNR-EXT dataset contains images with sunny days, cloudy days, rainy days, and different dates and times. There is a sufficient number of images for training. The model was set for training in 10 periods, with a batch size of 128 and a learning rate of 0.0001.

How does this run the code for the implementation method：

# building network

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras import optimizers

from keras.preprocessing.image import ImageDataGenerator

import time

model = tf.keras.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dropout(0.5, noise\_shape = None, seed = None))

model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy',

optimizer=optimizers.RMSprop(lr=1e-4),

metrics=['acc'])

model.summary()

train\_dir = "./datasets/CNR-EXT-Patches-150x150/train"

validation\_dir = "./datasets/CNR-EXT-Patches-150x150/val"

test\_dir = "./datasets/CNR-EXT-Patches-150x150/test"

# All images will be rescaled by 1./255

train\_datagen = ImageDataGenerator(rescale=1./255)

validation\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

# This is the target directory

train\_dir,

# All images will be resized to 150x150

target\_size=(150, 150), #(75, 75),

batch\_size=128,#20

## Improve Classification Model

Same as Chapter 6.3, To train the Improve classification models, CNR-EXT data sets were used. 94,493 images were extracted from the database as the training set of the model. The CNR-EXT dataset contains images with sunny days, cloudy days, rainy days, and different dates and times. There is a sufficient number of images for training. The model was set for training in 10 periods, with a batch size of 128 and a learning rate of 0.0001.

How does this run the code for the implementation method：

# building network

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras import optimizers

from keras.preprocessing.image import ImageDataGenerator

import time

model = tf.keras.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)))

#input\_shape=(150, 150, 3)))

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.ZeroPadding2D(padding=(1, 1), data\_format=None))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.ZeroPadding2D(padding=(1, 1), data\_format=None))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.BatchNormalization())

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dropout(0.5, noise\_shape = None, seed = None))

model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary\_crossentropy',

optimizer=optimizers.RMSprop(lr=1e-4),

metrics=['acc'])

model.summary()

train\_dir = "./datasets/CNR-EXT-Patches-150x150/train"

validation\_dir = "./datasets/CNR-EXT-Patches-150x150/val"

test\_dir = "./datasets/CNR-EXT-Patches-150x150/test"

# All images will be rescaled by 1./255

train\_datagen = ImageDataGenerator(rescale=1./255)

validation\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

# This is the target directory

train\_dir,

# All images will be resized to 150x150

target\_size=(150, 150), #(75, 75),

batch\_size=128,#20

# Since we use binary\_crossentropy loss, we need binary labels

class\_mode='binary')

validation\_generator = validation\_datagen.flow\_from\_directory(

validation\_dir,

target\_size=(150, 150), #(75, 75),

batch\_size=128,#20

class\_mode='binary')

# Perform the training (fit the model with data)

starttime = time.time()

history = model.fit(

train\_generator,

steps\_per\_epoch=739, # 5000/20=250 #94493/128=739 489364/128=3824 96493/128

epochs=10, #30, 10

validation\_data=validation\_generator,

validation\_steps=146) # 1000/20=50 #18647/128=146

print(str((time.time()-starttime)/60) + ' (min)')

# Save model

model\_name = 'CNR\_MODEL.h5'

model.save(model\_name)

## Displaying Result

After the model is established and properly trained, the model can be put into the parking lot detection system for classification detection. The pre-cut image patches are passed to the classification model, which will predict the results of the occupation status and display the results in the image. In the image display, red rectangles are used to mark BUSY parking Spaces and green rectangles are used to mark free parking Spaces.

This is the implementation for displaying result code.

# Test the Model in testing dataset

import pandas as pd

import tensorflow as tf

import cv2

import sys

import os

import numpy as np

import time

from matplotlib import pyplot as plt

from PIL import Image

import csv

from tensorflow.keras import models

start = time.time()

count = 0

path\_source = './modelcar/'

path\_location = 'modelcar\_cam.csv'

patch\_size = (150,150)

font = cv2.FONT\_HERSHEY\_COMPLEX # for better font display

# construct "bbox\_set2" dict

bbox\_set2={}

with open(path\_location) as csvfile:

reader = csv.reader(csvfile)

#for line in reader:

for idx, line in enumerate(reader):

if idx==0:

continue

# add into bbox\_set

bbox\_set2[int(line[0])]= [int(line[1]), int(line[2]), int(line[3]), int(line[4])]

# Perform classification for each parking lot

for files in os.listdir(path\_source):

if files.endswith('.JPG'):

image = path\_source + files

# Perform classification for each parking lot

img = cv2.imread(image)

patches = {} # create a dictory to store all the patches

# extract patches

for slotId, bbox in bbox\_set2.items():

bbox\_set3 = (np.asarray(bbox)).astype(int)

patch = img[bbox\_set3[1]:bbox\_set3[1]+bbox\_set3[3], bbox\_set3[0]:bbox\_set3[0]+bbox\_set3[2]]

patch = cv2.resize(patch, patch\_size)

# append patch into patches dict

patches[slotId] = patch

# convert to RGB format, compatible with keras training

patch\_input\_rgb = cv2.cvtColor(patch, cv2.COLOR\_BGR2RGB)

patch\_input\_rgb = patch\_input\_rgb/255.0 # rescale to (0,1) range

patch\_tensor = np.expand\_dims(patch\_input\_rgb, axis=0)

#Let patch\_input be forwarded through the networ3

last\_output\_model = models.Model(inputs=model.input, outputs=model.layers[-1].output)

output = last\_output\_model.predict(patch\_tensor)

if output[0,0] > 0.5:

output\_str\_local = "FREE"

color = (0,255,0)

else:

output\_str\_local = "BUSY"

color = (0,0,255)

output\_str\_local=str(slotId)+":"+output\_str\_local

# Put the ID text at the top of the Bbox

cv2.putText(patch, output\_str\_local, (0, patch.shape[1]//2),

font, bbox\_set3[2]/100.0 , (0,255,255), 2)

Draw the final output (busy or free) to the img

cv2.rectangle(img, (bbox\_set3[0], bbox\_set3[1]), (bbox\_set3[0]+bbox\_set3[2], bbox\_set3[1]+bbox\_set3[3]), color,3)

# Put the ID text at the top of the Bbox

cv2.putText(img, str(slotId),

(bbox\_set3[0]+bbox\_set3[2]//2-bbox\_set3[2]//4, bbox\_set3[1]+bbox\_set3[3]//2),

font, bbox\_set3[2]/100.0 , (0,255,255), 2)

# save img

count += 1

outputimg = './test123/output' + str(count) + '.jpg'

cv2.imwrite(outputimg, img)

print(str(outputimg) + ' is save! The time is: ' + str(time.time() - start))

# display img

img\_display = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

plt.imshow(img\_display)

plt.show()

A picture containing indoor, table, sitting, computer

Description automatically generated

Figure 5: Sample Detection Result with model car

# System Quality

## Past Classification model

The past classification model was trained using CNR-EXT. In order to evaluate the effectiveness of the classification model, two different data sets, images of CNR-EXT and PKLot, were used for the test set. Before the test, the 31,825 images of CNR-EXT were used as the test set. The images used for training must be different from the images used for testing so that the test set would be useful. All 462,856 images of PKLot were used as the test set, and then PKLok's bright 179,886 images and dim 282,990 images were separated into two test sets. During the training, the model will continue to learn. In general, as the number of epoch increases, the accuracy will increase and the loss will decrease.

After ten periods of training, the CNR-EXT accuracy and loss of the training data are 0.9912 and 0.0281, respectively.

The CNR-EXT accuracy and loss of verification data are 0.9832 and 0.0572 respectively.

The CNR-EXT accuracy of the test data is 0.9890 respectively.

The PKLot accuracy of the test data is 0.6955 respectively.

The PKLot(Bright) accuracy of the test data is 0.5245 respectively.

The PKLot(Dim) accuracy of the test data is 0.8049 respectively.

A screenshot of a cell phone

Description automatically generated

Figure 6: Past Classification Model training and validation accuracy

A screenshot of a cell phone

Description automatically generated

Figure 7: Past Classification Model training and validation loss

Table 5: Summary of experiment result (Past Classification Model)

|  |  |
| --- | --- |
|  | Accuracy |
| CNR-EXT(training) | 0.9912 |
| CNR-EXT(validation) | 0.9832 |
| CNR-EXT(testing) | 0.9890 |
| PKLot(testing) | 0.6955 |
| PKLot(Bright) | 0.5245 |
| PKLot(Dim) | 0.8049 |

## Improve Classification model

The Improve classification model was trained using cnr-ext. In order to evaluate the effectiveness of the classification model, two different data sets, images of cnr-ext and PKLot, were used for the test set. Before the test, the 31,825 images of cnr-ext were used as the test set. The images used for training must be different from the images used for testing so that the test set would be useful. All 462,856 images of PKLot were used as the test set, and then PKLok's bright 179,886 images and dim 282,990 images were separated into two test sets. During the training, the model will continue to learn. In general, as the number of epoch increases, the accuracy will increase and the loss will decrease.

Finally, the CNR-EXT accuracy and loss of the training data are 0.9977 and 0.0076 respectively.

The CNR-EXT accuracy and loss of verification data are 0.9855 and 0.0871 respectively.

The CNR-EXT accuracy of the test data is 0.9801 respectively.

The PKLot accuracy of the test data is 0.8863 respectively.

The PKLot(Bright) accuracy of the test data is 0.7834 respectively.

The PKLot(Dim) accuracy of the test data is 0.9536 respectively.

A screenshot of a cell phone

Description automatically generated

Figure 8: Improve Classification Model training and validation accuracy

A screenshot of a cell phone

Description automatically generated

Figure 9: Improve Classification Model training and validation loss

Table 6: Summary of experiment result (Improve Classification Model)

|  |  |
| --- | --- |
|  | Accuracy |
| CNR-EXT(training) | 0.9978 |
| CNR-EXT(validation) | 0.9855 |
| CNR-EXT(testing) | 0.9801 |
| PKLot(testing) | 0.8863 |
| PKLot(Bright) | 0.7834 |
| PKLot(Dim) | 0.9536 |

## Compare accuracy of two models

According to the following table, we can see that the accuracy of the Improve Classification Model in the original CNR-EXT test set has decreased by 0.89%, but it has increased by 19.09% in the PKLot test, and the PKLot data with a brightness value greater than 150 Incorporated, it has increased by 25.89%. It is proved from experiments that the addition of the normalization method can reduce the overfitting of the model, thereby enhancing the Robust of the model

Table 7: Compare Accuracy of two models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | CNR-EXT (training) | CNR-EXT (validation) | CNR-EXT (testing) | PKLot (testing) | PKLot (Bright) | PKLot (Dim) |
| Past Classification Model | 0.9912 | 0.9832 | 0.9890 | 0.6955 | 0.5245 | 0.8049 |
| Improve Classification Model | 0.9978 | 0.9855 | 0.9801 | 0.8863 | 0.7834 | 0.9536 |
|  | +0.0066 | +0.0023 | -0.0089 | +0.1909 | +0.2589 | +0.1487 |

## Check for irregular parking

From the Figure, we can find that for the vehicles that do not follow the ground line of the parking space, the two cars at the bottom of the figure can successfully identify the occupancy as BUSY, while the private cars and buses above can only correctly recognize that one of them is BUSY, The grid detects errors, and the irregular parking system needs to be improved. Although the training set is all pictures of private cars, it can also identify motorcycles normally, proving that the model has a certain generalization ability.

A stack of flyers on a table

Description automatically generated

Figure 10: irregular parking sample

# Ethics and Professional Conduct

Macau currently has the relevant laws of the Personal Data Protection Law. It is suitable for processing personal data in whole or in part by automatic means. [13] The outdoor intelligent parking lot detection system mainly uses cameras to capture images to detect parking spaces. When collecting images of parking spaces, you can collect a lot of personal information such as license plates, driver, or passenger’s faces. This may lead to violations of personal privacy and may violate Macau laws. Therefore, the relevant departments should be consulted before the system is officially used.

And even if you collect images with the consent of others, you should pay attention to the storage location of the image data. According to the Personal Data Protection Act, personal data is any information that can identify a person, including sound and images. If the system data access server is not in Macau and involves the transfer of personal data to areas outside Macau, you may need to notify the relevant department or apply for a license.

As for data protection, because the current camera resolution can easily clap hands to take 4K images, the image is very clear. If you show the image to the public, there will be certain privacy issues. Anyone with interest can use the system to get a certain The whereabouts of people, so the image data should depend on the situation and decide whether to store high-definition pictures.

Therefore, outdoor parking systems should ensure that all high-resolution images are not stolen by hackers. For example, set up a firewall about the server, delete hd images or check that the system is not connected to an external network.

Add security protection to the camera system as soon as the real-time status of each parking space is detected. This prevents others from stealing hd images directly from the camera system.

Finally, if any personal information is involved, the Macao Personal Data Protection Office is responsible for monitoring and coordinating the Personal Data Protection Act. For example, if a camera lens is installed in a public place and can capture the location of others, it is best to consult the office first, explain the purpose of the installation, and seek help. This helps prevent the system from violating Macao law.

# Summary

The report shows how to use convolutional neural network model to generate the outdoor parking lot detection system, the advantages of using convolutional neural network for image detection, and how to use regularization method to enhance the robustness of the model.

We used the CNR-ext database to train the basic Classification model. In the trained model, we use images from the same dataset to test the model's performance. Experimental results show that the accuracy of the model is 99% in the original data set. This is due to sufficient training data and test data from the same parking lot. This proves that CNN is very suitable for image detection and can be used as a method to detect parking space occupancy.

To test the generalization ability of this model, the PKLot data set is used. The results show that the accuracy of the model in PKLot is not as good as that of CNR-ext due to the different environment, shooting Angle and time of the two parking lots. In order to better understand the reason, after dividing PKLot data into light and dark test sets, it was found that the image with sufficient daylight was less accurate and could accept normal or slightly dark images.

This is because there are fewer overexposed images in CNR-Ext. To solve this problem, regularization methods have been added to the classification model to reduce the overfitting of the model. Practical results show that with BatchNormalization, this type of image has been greatly improved. This also suggests that using regularization can help improve accuracy. To further improve the accuracy, some images can also be extracted from PKLot to train the model.

Finally, there is a test run model that simply checks the robustness of the model. As a result, you say that the model still needs to be improved for irregular parking, but it can effectively identify which type of car is parked in a parking space.

# References

[1] statistics and census service. [online]. available: <https://www.dsec.gov.mo/zh-MO/>

[2] CNR Parking Dataset - Dataset for visual occupancy detection of parking lots. [Online]. Available:

http://cnrpark.it/.

[3] “Parking Lot Database,” Parking Lot Database – Laboratório Visão Robótica e Imagem. [Online].

Available: https://web.inf.ufpr.br/vri/databases/parking-lot-database/.

[4] Goodfellow, Y. Bengio, and A. Courville, Deep Learning (Adaptive Computation and Machine

Learning series). The MIT Press, 2016.

[5] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image

Recognition, Sep. 2014.

[6] Nielsen, Michael A. *Neural networks and deep learning*. Vol. 2018. San Francisco, CA, USA:: Determination press, 2015.

[7] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[8] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[9] “Numpy”, Numpy. [Online]. Available: <https://numpy.org/>

[10] “Keras: The Python Deep Learning library,” Home - Keras Documentation. [Online]. Available:

https://keras.io/.

[11] “OpenCV,” OpenCV. [Online]. Available: <https://opencv.org/>.

[] Anaconda. [Online]. Availabe: <https://www.anaconda.com/>

[13]Office for Personal Data Protection. [Online]. Available: https://www.gpdp.gov.mo/.