



北京邮电大学



Queen Mary
University of London

Undergraduate Project Report 2018/19

Automatic Identification of Ships on Satellite Images

| | |
|-------------------|-----------------------------------|
| Name: | Zhaoxiong Chen |
| School: | International School |
| Class: | 2015215121 |
| QM Student No.: | 151007198 |
| BUPT Student No.: | 2015213445 |
| Programme: | Internet of Things Engineering |

Date: 06-05-2019

Table of Contents

| | |
|--|-----------|
| Abstract | 3 |
| Chapter 1: Introduction..... | 5 |
| 1.1 Project Motivation | 5 |
| 1.2 Project Summary | 5 |
| 1.3 Project Achievement | 6 |
| 1.3.1 Research | 6 |
| 1.3.2 Application | 6 |
| 1.3.3 Conclusion from Comparison..... | 6 |
| 1.3.4 Report | 7 |
| Chapter 2: Background | 8 |
| 2.1 Deep Learning Methodology..... | 8 |
| 2.1.1 Traditional Deep Learning..... | 8 |
| 2.1.2 Transfer Learning | 8 |
| 2.2 Mechanism of Neural Network in Nutshell | 8 |
| 2.2.1 Logistic Regression | 8 |
| 2.2.2 Structure of Node in Neural Network..... | 9 |
| 2.2.3 Activation Functions..... | 9 |
| 2.2.4 Cost Function..... | 10 |
| 2.2.5 Gradient Descent | 10 |
| 2.2.6 Forward and Backward Propagation..... | 10 |
| 2.2.7 Epoch and Batch Size..... | 11 |
| 2.3 Convolutional Neural Network..... | 11 |
| 2.3.1 Overview | 11 |
| 2.3.2 Convolutional Layer | 12 |
| 2.3.3 Pooling Layer | 13 |
| 2.3.4 Fully-Connected Layer | 14 |
| Chapter 3: Design and Implementation | 15 |
| 3.1 Research Methodology | 15 |
| 3.1.1 Early Stage..... | 15 |
| 3.1.2 Main Stage..... | 15 |
| 3.1.3 Later Stage..... | 16 |
| 3.2 Data Used in Application..... | 16 |
| 3.2.1 Airbus Dataset | 16 |
| 3.2.2 Pre-Trained Parameters | 17 |
| 3.3 Application Environment and Dependencies..... | 18 |
| 3.3.1 Hardware Environment..... | 18 |
| 3.3.2 Software Dependencies | 18 |
| 3.4 Application Organisation | 19 |
| 3.4.1 Package “Analysis” | 20 |
| 3.4.2 Package “DataPrep” | 20 |
| 3.4.3 Package “Evaluation”..... | 24 |
| 3.4.4 Package “ModelsTraining”..... | 24 |
| 3.4.5 Package “Utilities” | 25 |
| 3.4.6 Default Package..... | 25 |
| Chapter 4: Results and Discussion | 26 |
| 4.1 Results | 26 |
| 4.1.1 Outcomes from Application..... | 26 |
| 4.1.2 Model Accuracy and Loss | 27 |

Automatic identification of ships on satellite images

| | |
|---|-----------|
| 4.1.3 Time Consumption of Model Training | 28 |
| 4.1.4 Model Performance Visualisation..... | 28 |
| 4.1.5 Ship Existence Prediction Visualisation | 32 |
| Chapter 5: Conclusion and Further Work..... | 33 |
| 5.1 Conclusion about Outcomes..... | 33 |
| 5.2 Conclusion about Model Performance and Training Time..... | 33 |
| 5.2.1 Model with the Best Performance | 33 |
| 5.2.2 Factors that Impact on Performance | 33 |
| 5.2.3 Factors that Impact on Training Time..... | 34 |
| 5.3 Problems and Solutions | 35 |
| 5.3.1 Configuration on Software Drivers | 35 |
| 5.3.2 Model Accuracy Improvement | 35 |
| 5.4 Project Redo Differences | 36 |
| 5.5 Further Work | 36 |
| 5.5.1 Ship Type Recognition | 36 |
| 5.5.2 Improve Accuracy by Combining Multiple Remote Sensing | 36 |
| 5.5.3 Advanced User-friendly Engineering | 36 |
| References | 37 |
| Acknowledgement | 38 |
| Appendix | 39 |
| I Python Environment Package Dependencies..... | 39 |
| II Architecture Summary of Implemented Models | 40 |
| Risk Assessment..... | 48 |
| Environmental Impact Assessment..... | 49 |

Abstract

Recently, shipping traffic has been growing rapidly and so are the frequent occurrence of problems start from environmental disasters to malfeasances. Although people have developed various monitoring resorts like satellite imaging to ship monitoring by providing a lot of useful image source, it is a matter of fact that the analysis method of certain data source are not handy enough. There is still many information buried within the data source, which may lead to a waste in human or financial resources. This project tried to do a research in deep learning area, implement four kinds of remarkable architectures, simply modify some of the models, train models in different approaches, compare the performance of implemented models and bring visualisation to as many deep learning stages as possible. Various phenomenal models are trained in this project and some of those models can reach up to 93% accuracy. Aspects from data preparation, model architecture, model training history to prediction on inputted images are visualised. Due to the time limitation, the project still possesses some space of improvement, such as further improvement on accuracy, deeper exploration on hyperparameter, architecture and more advanced feature e.g. graphic user interface to be developed. In a word, this project successfully achieved its designed goals, which are model implementation, model performance comparison and visualisation.

Keywords - Image processing, machine learning, deep learning

摘要

近年来，航运量一直处于快速增长的状态。同时一系列从环境灾难到违法行为的问题也时常发生。尽管人们已经开发了各种监控方式，如通过提供大量有用的图像源来进行监控的卫星成像，但事实上这些数据源的分析方法不够方便。这些数据中仍然有大量为提取出来的信息，这可能会导致人力或财力资源的浪费。本项目在深度学习领域进行了简要研究，试图完成四种业界公认的图像识别架构，简要修改模型，使用不同方法训练模型，比较模型的性能，并尽可能地将可视化带入到深度学习的各个阶段。本项目运行了一些出色的模型，其中一些模型可以达到高达 93% 的准确率。从数据准备，模型架构设计，模型训练结果到输入图片预测，本项目成功进行了可视化。由于时间的限制，该项目仍然具有一定的改进空间，如进一步提高准确性，深入探索超参数和开发更高级的功能，如图形用户界面。总而言之，本项目成功实现了之前设计的目标，即模型实现，模型性能比较以及可视化。

关键词-图像处理，机器学习，深度学习

Chapter 1: Introduction

1.1 Project Motivation

With the development of economy and commercial, it can be witnessed a dramatic growth on the transportation, especially on the international shipping traffic. However, the bloom of shipping traffic has increased the risk from accidents causing environmental hazard to infraction manners like piracy on public sea and illegal whales fishing. The risks have brought disorder to cargo transportation, thus damaged the profit in cargo industries. More importantly, the risks could discourage investors and eventually have a bad influence on global economy.

In order to lower the risk of such accidents mentioned above, there is an urgent need of ship traffic monitoring. Luckily, a growing number of satellite images are collected through remote sensing technology. More satellites have been set up to surveillance service. However, the lack of methodology makes the images not capable of being fully exploited, leaving huge amount of information buried under the ocean of images.

Fortunately, the renaissance of machine learning, started from 2010s, have brought this problem a novel solution. By using deep learning technologies computers are capable to make predictions toward the existence of ship from satellite images.

1.2 Project Summary

The project aims to implement state-of-the-art data science (deep learning) approaches to correctly identify ships from satellite images. A dataset from Airbus will be used to help develop and improve data science methods.

From my point of view, benefits of implementing this project could be the huge potential profit toward those giant cargo companies, because they are the target customers of this project who always sensitive to costs.

By finding more sensible, intuitive and highly-scalable methods to identify ships in satellite images, companies would monitor ships faster, lower the cost of specialised staffs who take charge of ship surveillance and make the software upgrade easier. By helping cargo enterprises anticipate threats and optimize traffic schedule, the project would help the company decrease the risk of accident and then the cost on compensating. Moreover, optimization on scheduling could improve the operational efficiency and make more profits. In a word, the project could help cut down the budget while enhance profits of business operation.

1.3 Project Achievement

1.3.1 Research

Research is essential to almost every project. To finish this project, papers about various neural network, documents of Python packages, books and online courses concerning deep learning are covered in research scope.

1.3.2 Application

As an implementation project, functionalities of software project are the very essences. Here are the several key features of the system:

- **Recognition of ship through the satellite image**

Using convolutional neural network (CNN), the system could provide prediction to a bunch of input image about the existence of ships with relatively high accuracy.

- **Analysis on performance and visualisation**

Importing relevant Python packages, the system is capable to generate bar charts for dataset statistics, architecture of trained model, outcome of data augmentation. More importantly, it can also analyse the performance by plotting line graph of model “loss” and “accuracy” according to training history.

- **Integral framework for neophytes**

After implementation, this project is equipped with a series of deep learning operation from data analysis, pre-processing to model training and evaluation. It could be treated as a “framework” for deep learning area neophytes for its clear, reasonable structure and detailed documentation. It is believed that this project will make the deep learning development easier.

1.3.3 Conclusion from Comparison

This project delivers various conclusion through comparison between implemented data science methods. This project tried to draw conclusion from these three aspects:

- Performance of different CNN on same dataset, same backend, same epoch, same batch size;
- Performance of different CNN on same dataset, same backend, same epoch, different batch size;

- Performance of nearly same CNN with different, slightly modification on same dataset, same backend, same epoch, same batch size;

1.3.4 Report

Report is also an unignorable section which summaries the whole project. Here is the report organisation:

In “Chapter 1: Introduction”, motivation and expected benefit of this project is discussed. Furthermore, achievements are presented briefly.

In “Chapter 2: Background”, the technical context of this project is introduced to make project clearer to people.

In “Chapter 3: Design and Implementation”, the report delivers detail of research methodology, dataset used, environment and organisation of the software application. Application organisation is demonstrated in package sequence.

In “Chapter 4: Result and Discussion”, the performance figure and visualisation of implemented deep learning method is introduced.

In “Chapter 5: Conclusion and Further Work”, some crispy conclusions and future prospect of project development can be seen in this section.

Chapter 2: Background

2.1 Deep Learning Methodology

2.1.1 Traditional Deep Learning

To begin with, technology that this project based on is called deep learning. According to the definition from Investopedia.com, deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in AI that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network (Hargrave, 2019).

2.1.2 Transfer Learning

Transfer learning is one of the most distinguished deep learning branches. An essay published on machinelearningmastery.com said that it is a technique where a model trained on one task is re-purposed on a second related task. In other words, it is an optimization that allows rapid progress or improved performance when modelling the second task. In transfer learning, pre-trained models are used as the starting point on tasks like computer vision given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems. It is widely exploited given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained. Due to its characteristic, transfer learning only works well if the model features learned from the first task are general (Brownlee, 2017).

2.2 Mechanism of Neural Network in Nutshell

2.2.1 Logistic Regression

Logistic regression is probably the core to every deep learning problems. In “Deep Learning Specialization”, Andrew Ng said that logistic regression is a learning algorithm used in a supervised learning problem when the output y are all either zero or one. The goal of logistic regression is to minimise the error between its predictions and training data. In any binary detection problem, including “ship detection”, this algorithm will evaluate the probability of the object detected being in that image, which is described in equation (1) below (Ng, 2018). Cost function, mentioned later in detail, can minimise error or deviation.

$$\text{Given } x, \hat{y} = P(y = 1|x), \text{ where } 0 \leq \hat{y} \leq 1 \quad (1)$$

Linear regression is the counterpart to logistic regression. The reason why neural network chooses logistic regression rather than linear regression is that neural network is the solution to classification problems whereas linear regression is the solution to continuous value prediction as known as “regression problems”.

2.2.2 Structure of Node in Neural Network

Node is basic unit in neural network which imitates genuine neural cell in human brain. A neural network is exactly the combination of nodes.

Figure 2.1 represents a node in neural network. Each node receives the output values from previous layer and input them to certain layer for further calculation. After calculation, the node outputs result to be fed in next layer. In neural network problems, output can be concluded as $\hat{y} = \sigma(w^T x + b)$, where σ is activation function, w is the weight of each input to neuron node, x is input values and b is bias of each output.

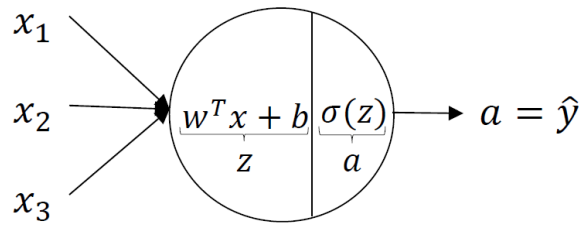


Figure 2.1 Neural Network Node Representation

2.2.3 Activation Functions

Activation function is the application of logistic regression in neural network problems. In recent years several activation functions are introduced such as “Sigmoid”, “tanh”, “Rectified Linear Unit (ReLU)”, “Softmax” and “Leaky-ReLU”. In this project, “Sigmoid” and “ReLU” are two widely functions in models training. Equation (2) and (3) are mathematical representation of “Sigmoid” and “ReLU” function while Figure 2.2 are their graphical representation.

$$\text{Sigmoid: } a = \frac{1}{1 + e^{-z}} \quad (2)$$

$$\text{ReLU: } a = \max(0, z) \quad (3)$$

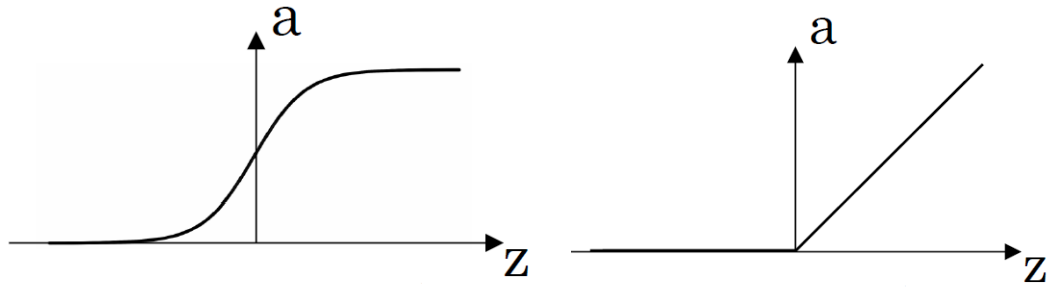


Figure 2.2 Graphical Representation of "Sigmoid" (left) and "ReLU" (right) Functions

2.2.4 Cost Function

Since the goal of logistic regression is to try to minimise the deviation between predicted values and real values, it is necessary to have a clear mathematical representation of this problem. Cost function or loss function is the equation that calculates the discrepancy between predicted and actual values. According to Ng, lost function computes the error for a single training example while cost function is the average to the loss function of the entire training set. The target is to minimise the overall cost function using proper parameters weight w and bias b .

Two common forms of cost function are “Mean Squared Error” used in linear regression and “Cross Entropy” in logistic regression. Equation (4) is the mathematical representation of “Cross Entropy” cost function.

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m [(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \quad (4)$$

2.2.5 Gradient Descent

According to the definition from ML Cheatsheet, gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, gradient descent could be used to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks (ML Cheatsheet, 2017). Gradient descent is the approach to lower cost function $J(w, b)$.

2.2.6 Forward and Backward Propagation

Forward and backward propagation are two algorithms that are often mentioned in deep learning field. Forward propagation algorithm is the action of calculating prediction result using current weight w and bias b , whereas backward propagation algorithm is the action of

adjusting weight and bias parameter w' and b' used in previous training epoch based on loss function result.

2.2.7 Epoch and Batch Size

In actual training process, there are two hyperparameters that need to be investigated thoroughly: training epoch and batch size. Epoch is the times of model training upon same training set. Batch size is the number of images popped into the model at each step. Step number in each epoch can be calculated as equation (5).

$$\#Step = \frac{\#Images\ in\ Training\ Set}{\#Batch\ Size} \quad (5)$$

2.3 Convolutional Neural Network

2.3.1 Overview

CNN is so far the mainstream method in solving computer vision problems. Article on techtarget.com concludes that it is a type of artificial neural network that is specifically designed to process pixel data. It is widely used in tackling image recognition and processing problems like object recognition, human face recognition and so on (Rouse, 2018).

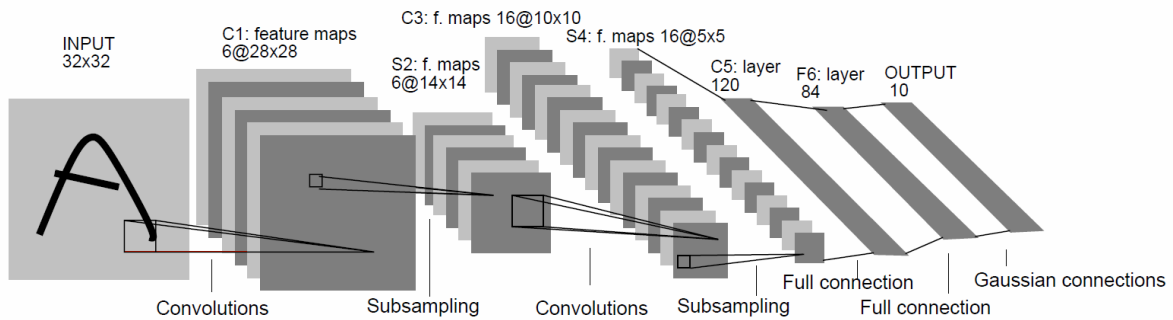


Figure 2.3 Architecture of LeNet-5

Figure 2.3 above is the architecture of LeNet-5, a world-famous hand writing number recognition method (LeCun, Bottou, Yoshua, & Haffner, 1998). It inputs a 32×32 monocoloured image and outputs prediction of actual hand-written number. It can be observed that LeNet-5 is a typical CNN that consists of a few of convolutional layer, subsampling (pooling) layer and fully-connected layer, which is CONV-POOL-CONV-POOL-FC-FC-FC.

In CNN, image matrix that recording pixel values of specific image is firstly convoluted by filters. It is possible that an image will be convoluted several times continuously. Next, the

convoluted image matrix will be inputted into pooling layer. Then, the matrix is more likely to be inserted into several combinations of convolutional layers and pooling layers, i.e. N Convolutional Layers + 1 Pooling Layer. Finally, matrix is flattened and derived into more than one fully-connected layer (or dense layer). Prediction of certain image in probability will be outputted in the last dense layer.

2.3.2 Convolutional Layer

- **Filter**

Figure 2.4 from CS231n course of Stanford University demonstrates the procedure of convolution operation (Stanford University, 2017). It can be witnessed that convolution function called filter (or kernel) will walk through the image matrix in zig-zag every N units, where N is stride defined by programmer. Each filter has its configuration that stands for a feature expected to be detected. For instance, stride is defined as 1 unit and filter $W1[:, :, 1]$ is more likely to detect a T-shaped pattern.

- **Padding**

Filters cannot always cover every pixel especially for the one in corners. Thus, padding is implemented to add p units of row and column in order to make all the pixels have equal chance to be convoluted. Two common mode of padding are widely used: one is “valid” means there is no padding while the other is “same” means padding will make output size the same as the input size.

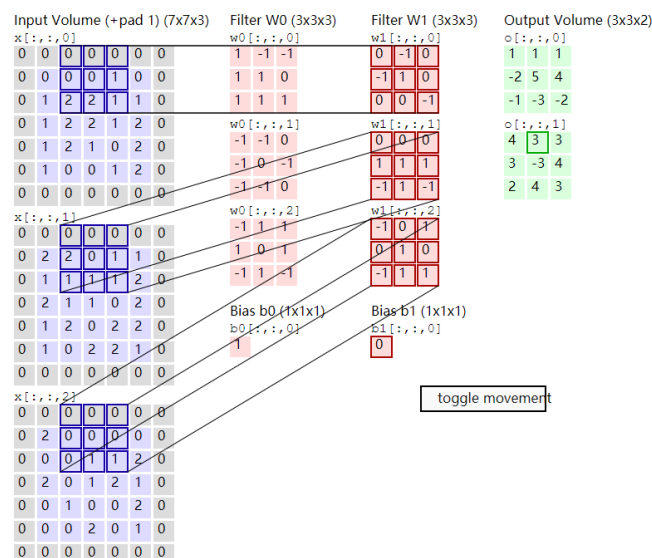


Figure 2.4 Convolution Operation on Input Images

From figure above it could be witnessed that after convolution by filter W0 and W1, shape

of input volume is switched from $7 \times 7 \times 3$ to $3 \times 3 \times 2$. Size of matrix after convolution can be calculated in equation (6) below, where image scale $n \times n$, filter scale $f \times f$ with padding p units on each side and stride is defined as s units.

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \quad (6)$$

After convolution, there will be an increase on depth of the matrix as Figure 2.5 shows. Weight of generated matrix is equivalent to number of filters in convolutional layer.

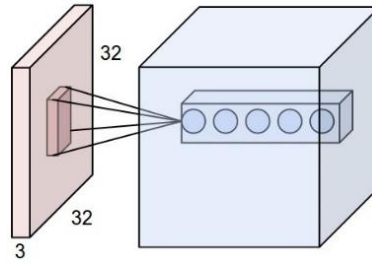


Figure 2.5 Depth Increasing after Convolution

● Activation Function

Activation function is concatenated after convolution. The most common used activation function in CNN is “ReLU”. More details are introduced above in “2.2.3 Activation Functions” section.

2.3.3 Pooling Layer

Pooling layer (also called down sampling layer) is assigned to contract the image feature. By replacing certain $f \times f$ area with its max or mean value, feature will be condensed and easier to be detected. Figure 2.6 is a typical maxpooling layer with filter sized 2×2 and move in zig-zag every 2 units.

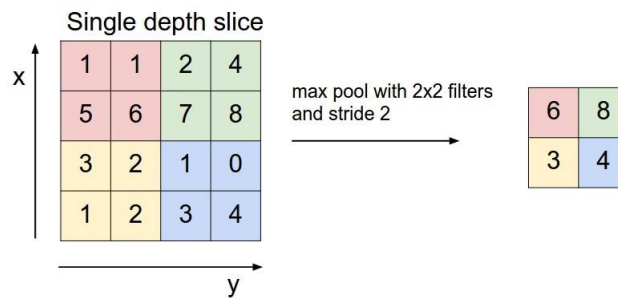


Figure 2.6 MaxPooling Layer

2.3.4 Fully-Connected Layer

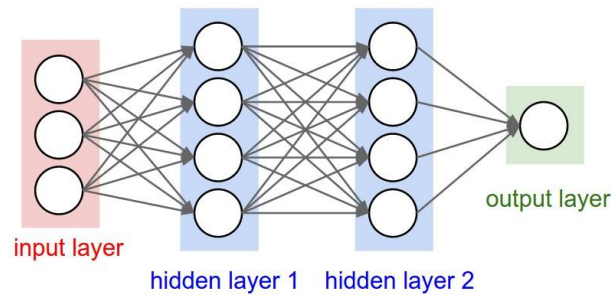


Figure 2.7 Structure of Neural Network/Dense Layer

Cluster of fully-connected (FC) layer (or dense layer) is traditional neural network according to Figure 2.7 above. Input FC layer in red receives fully-convoluted and flattened data matrix. When several cascaded hidden FC layers finished calculation, the prediction result will be generated through the final output FC layer in green.

Chapter 3: Design and Implementation

3.1 Research Methodology

The sequence of finishing the tasks are basically researching through the entire project, doing preparation work in early stage, developing the system in main stage and end up with collecting results in later stage.

3.1.1 Early Stage

Having a research and knowledge learning about deep learning to have a clear idea about deep learning region and obtain necessary information to start up the project. All these materials could be retrieved from books, research papers, community forum and online public classes like Coursera.com. Major aspects that covered in research and study are:

- Deep learning algorithms: e.g. Convoluted Neural Network;
- Deep learning frames: e.g. TensorFlow, Keras;
- Programming language: e.g. Python;
- Necessary software libraries: e.g. Python standard libraries and Scikit-image;
- Determining the algorithms that will be implemented in the project.

Apart from researching, exploration about dataset for this project was done. For thorough introduction of dataset, please refer to “3.2.1 Airbus Dataset” section. Apart from these datasets, no more data will be collected from other source for this project.

Eventually, preparation of deep learning environment was finished. Two PCs with Intel processor and Nvidia GPU were assigned and configured. For the software and hardware environment detail, please refer to “3.3 Application Environment and Dependencies” section.

3.1.2 Main Stage

Major task in main stage is the development of ship detection system. Agile method was followed, which encourages programmer to develop the project from a prototype in a gradual, iterative way. A wide range of alteration would be allowed, according to the Agile Manifesto. The iteration plan will be:

- Dividing dataset into several groups, including training set, validation set and testing set;
- Establishing certain algorithm on selected frame;

Automatic identification of ships on satellite images

- Code debugging;
- Develop analysis module according to kaggle.com to evaluate the performance of certain algorithm;
- Adjusting the system based on the performance feedback;
- Dataset augmentation;
- Learning further about deep learning;
- Other work not mentioned but should be done.

3.1.3 Later Stage

In later stage, performance data about training algorithm were collected. Also, some visualization methods were deployed to deliver the result in a more concrete way.

More importantly, Final Viva would be prepared in this stage. To guarantee the effect of demonstration, the system was altered to be more user-friendly by implementing a command line interface with exception handler.

3.2 Data Used in Application

3.2.1 Airbus Dataset

Dataset used in this project are provided by Airbus, the famous aircraft manufacture, and downloaded from kaggle.com, an online community of data scientists and machine learners owned by Google. Apart from datasets mentioned above, no more image data will be collected for this project.

The dataset includes 4 parts:

- Images for training: “train_v2”, containing 192,556 items;
- Images for testing: “test_v2”, containing 15,606 items;
- Data labelling file for training set: “train_ship_segmentations_v2.csv”;
- Blank data labelling file for Kaggle competition submission “sample_submission_v2.csv”.

In these two “comma separated values” (CSV) files, two columns “ImageId” and “EncodedPixels” stand for the name of each image and pixel number that encoding ships. In column “Encoded Pixels”, it can be found variety of integer pairs like “a b”. The number pairs provide the ground truth (in run-length encoding format) for the training images. For instance,

“a b” indicates that in pixel number “a” there are “b” consecutive pixels encoding ships. “EncodedPixels” column helps researchers to generate binary labels to images.

Using Python image processing library “scikit-image”, attributes of the training set and testing set are collected and logged in two CSV files “TrainingSetAnalysis.csv” and “TestSetAnalysis.csv” respectively. Explored attributes for each image are shown in Table 3-1.

Table 3-1 Analysed attributes of dataset

| Attribute | Description | Type of value | Value |
|--------------|----------------------|------------------|------------------------|
| Width | Width of image | Integer | 768 |
| Height | Height of image | Integer | 768 |
| #Channel | Number of channels | Integer | 3 |
| #Pixel | Number of pixels | Integer | 1769472 |
| MaxPixel | Maximum pixel value | Integer | Varies from 0 to 255 |
| MinPixel | Minimum pixel value | Integer | Varies from 0 to 255 |
| AveragePixel | Average pixel value | Integer | Varies from 0 to 255 |
| Type | Number of bits | Unsigned Integer | 8 |
| Extension | Type of image | String | “.jpg” |
| Size | Size of image (Byte) | Integer | Varies from each image |
| Directory | Directory of images | String | Varies from each image |

For “TrainingSetAnalysis.csv”, used to be converted into pandas DataFrame before feeding into model, it only records attributes that are necessary to the training. Therefore “Type”, “Extension” and “Size” columns are dropped. While in “TestSetAnalysis.csv” all of the characters except “Directory” listed on the table are recorded to present dataset attribute as much as possible.

3.2.2 Pre-Trained Parameters

With the progress of model training, it can be found that some of the CNN architecture obtains a huge number of parameters. For example, according to the research from Kaiming He, Residual Network with 50 layers (ResNet50) includes up to 158,857,089 parameters (He, Zhang, Ren, & Sun, 2016) while the 16-layered VGG16 network, introduced by Oxford Visual Geometry Group, has 134,264,641 parameters (Simonyan & Zisserman, 2014). It is inadequate to use all of 192,556 images to train models with such parameter scales, let alone the actual used sampled training set of 25,000 images. Therefore, transfer learning is implemented in the training of VGG16 and ResNet50. In transfer learning, pre-trained models are provided by Keras and trained using ImageNet project dataset.

3.3 Application Environment and Dependencies

3.3.1 Hardware Environment

Two PCs, a desktop and a laptop are involved in this project. Training jobs in project are entirely executed on the desktop while the demonstrating tasks are finished on the laptop.

Key specifications of computers are listed in the Table 3-2 below. They are believed to have a deep influence on model performance:

Table 3-2 Key Specification of PCs

| PC | Desktop (Training) | Laptop (Demonstrating) |
|----------------|-----------------------------|-----------------------------|
| CPU | Intel Core i7-6800K@3.40GHz | Intel Core i7-7700HQ@2.8GHz |
| GPU | NVIDIA GeForce GTX 1080Ti | NVIDIA GeForce GTX 1060 6GB |
| RAM | 16GB DDR4 2133 DIMM | 32GB DDR4 2400 SODIMM |
| Storage Device | HDD 2TB 5400RPM 64MB | HDD 1TB 5400RPM 128MB |

3.3.2 Software Dependencies

Software dependencies consists of three dimensions: OS/Drivers, Programming Language/IDE and specific package dependencies.

- **OS/Software drivers**

The project is developed on Windows 10 version 1803. Though it will take more effort in environment configuration on Windows than Linux, GPU driver support on Windows is way much better than that on Linux. To enable GPU acceleration on model fitting, NVIDIA CUDA driver 9.0 in associate with CUDA Deep Neural Network library (cuDNN) 7.4.1.5 are installed on desktop PC. Since visualization of model architecture also needs additional driver, package “Graphviz” is deployed. In addition, a module called “tee” transplanted from Linux to Windows is exploited to generate log file from command line.

- **Programming language/IDE**

This project chooses Python 3.6.8 64-bit as the programming language due to its flexibility on package management, which guarantees efficiency of project implementation. Plus, as a widely adopted deep learning language, there are uncountable solutions to various software bugs through Python coding. Thus, choosing it as deep learning programming language is beneficial to the developers. IDE employed in the project is PyCharm from JetBrains because of its powerful project management panel, user-friendly GUI and highly-

automated toolkits.

- **Specific package dependencies**

This project is based on mainly two deep learning backends: Keras 2.2.4 and Google's TensorFlow GPU 1.12.0, where Keras uses TensorFlow GPU as its backend. To visualise dataset statistics and model performance, Python package "matplotlib" is exploited. To give a concrete view of each model architecture, package "graphviz" and "pydot" are imported. Package "pandas" and "numpy" play important roles in managing dataset.

There are lots of supporting packages not mentioned here due to limitation on length. For detailed package dependency list, please refer to "Appendix I: Python Environment Package Dependencies".

3.4 Application Organisation

As Figure 3.1 shows below, the application is organised in 5 packages: "Analysis", "DataPrep", "Evaluation", "ModelsTraining" and "Utilities".

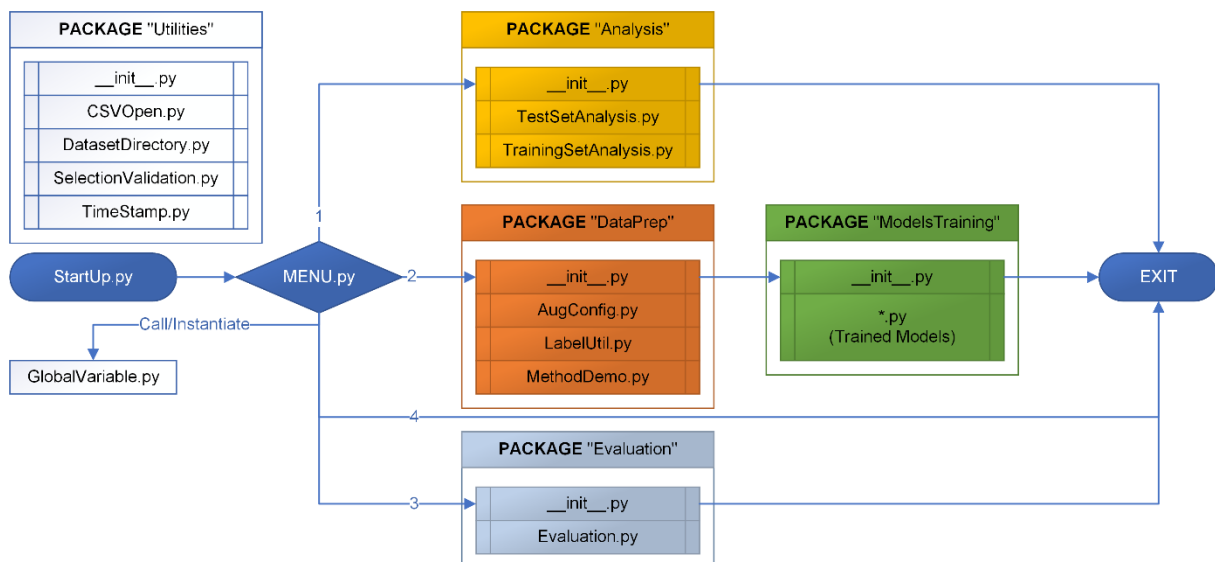


Figure 3.1 Application Structure and Relationship of Packages

3.4.1 Package “Analysis”

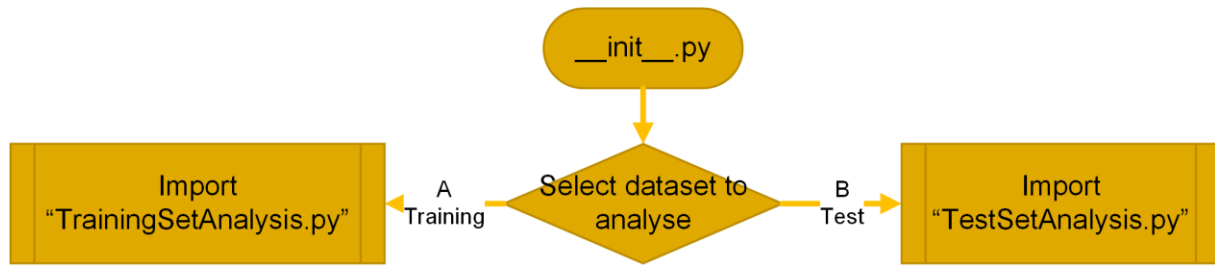


Figure 3.2 Flow chart of "Analysis__init__.py"



Figure 3.3 Flow chart of "TrainingSetAnalysis.py"



Figure 3.4 Flow chart of "TestSetAnalysis.py"

As Figure 3.2, 3.3 and 3.4 shows above, module “__init__.py” allows users to choose the dataset to be analysed. Module “TestSetAnalysis.py” and “TrainingSetAnalysis.py” generate CSV files that record image attributes (mentioned in 3.2.1 above) of testing set and training set respectively. The former module collects information by walking through the directory while the latter module follows the “ImageId” column in dataframe, which is converted from provided CSV file “train_ship_segmentations_v2.csv”.

3.4.2 Package “DataPrep”

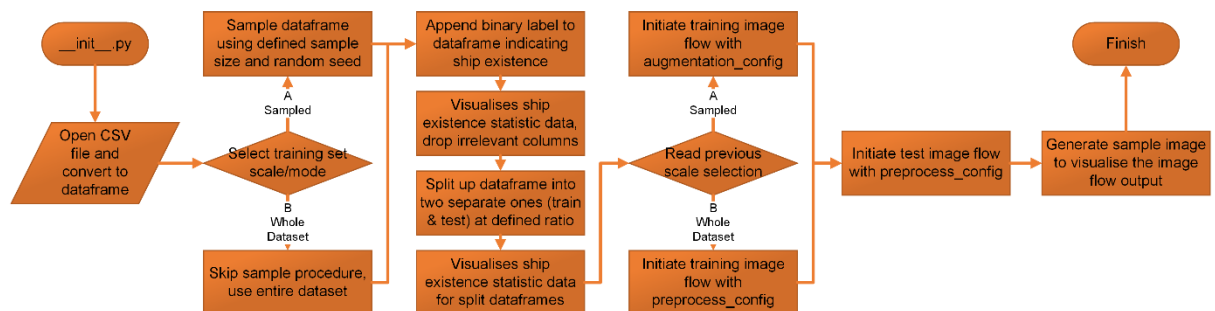


Figure 3.5 Flow chart of "DataPrep__init__.py"

Figure 3.5 above shows the procedure of module “__init__.py”. After reading CSV file and convert it into pandas dataframe, users can define the scale of dataset from two available options. “Sampled” indicates that system will only use arbitrary selected 25,000 images while “Whole DataSet” means that the entire training set of 192,556 images are used in model fitting. In this

Automatic identification of ships on satellite images

project, all models are trained under “Sampled” mode. Then, an image flow object called “DataFrameIterator” is initialised to feed image into model batch by batch. Finally, configuration about data augmentation is visualised through the module.

In “DataFrameIterator”, firstly it opens the actual image file using directory recorded on dataframe. Secondly, it converts image into numpy matrices of pixel values and pre-process certain image based on configuration. Thirdly, it generates processed pixel matrices that are readable to machine.

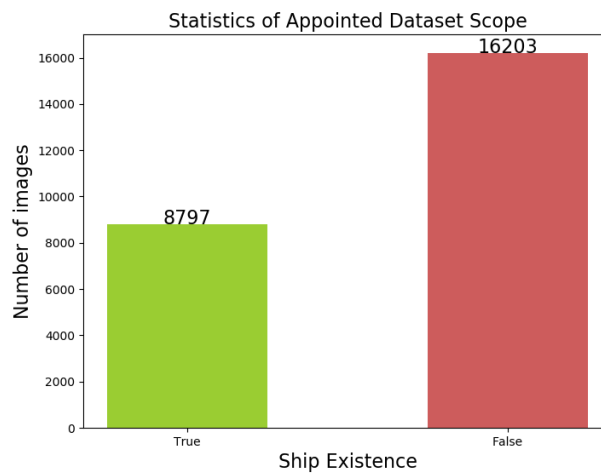


Figure 3.6 Statistics of Data at mode "Sampled"

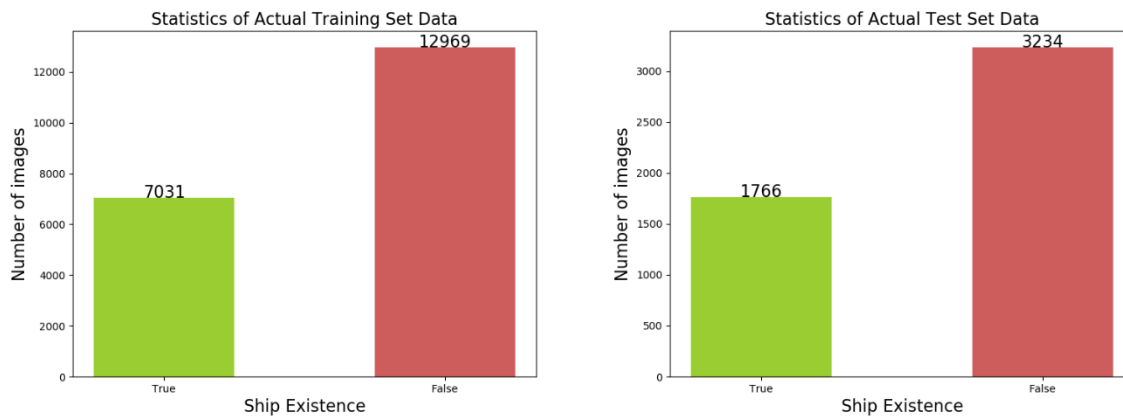


Figure 3.7 Statistics of Actual Training/Test Set Data at mode “Sampled”

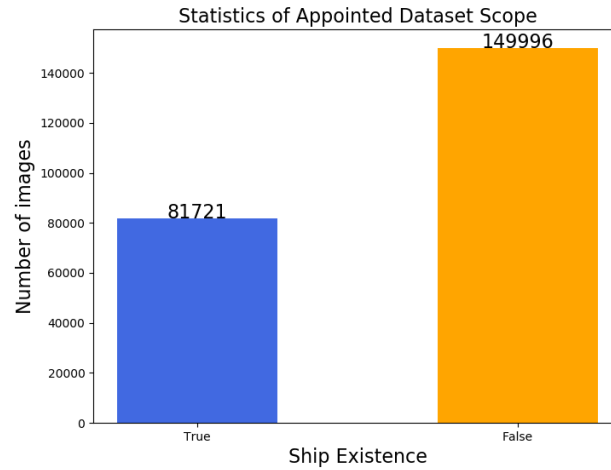


Figure 3.8 Statistics of Data at mode "Whole DataSet"

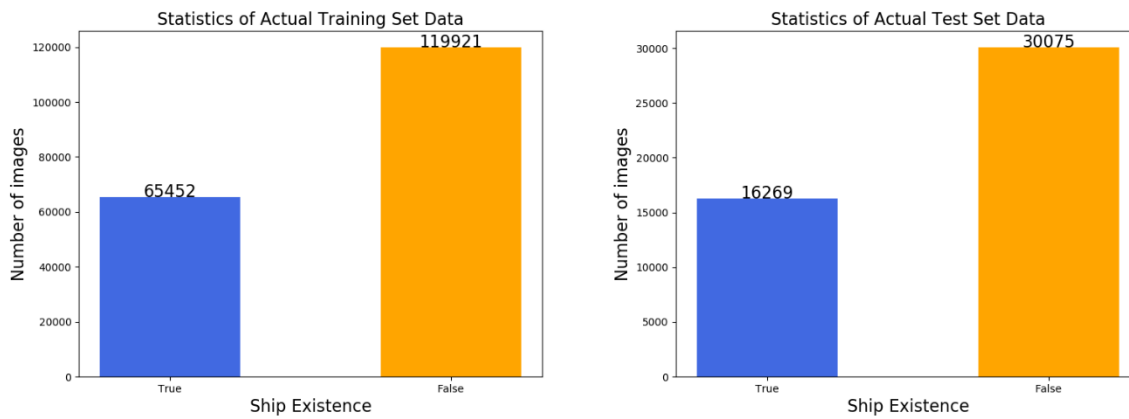


Figure 3.9 Statistics of Actual Training/Test Set Data at mode "Whole DataSet"

Module "AugConfig.py" defines data augmenting functions. The reason to augment data is to avoid overfitting from those ship images of same direction.

Module "LabelUtil.py" provides functions that can attach binary label to dataframe, calculate image number about ship existence and visualises the number.

Module "MethodDemo.py" defines a class that outputs the sample image using selected data augmentation setup. Figure 3.6, Figure 3.7, Figure 3.8 and Figure 3.9 above are six bar charts describing statistical data about ship existence in sampled and training/test set dataframe.



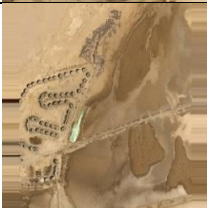

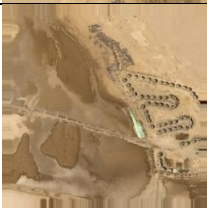

As for configuration detail of data augmentation, difference can be seen between these two modes. In "Sampled" mode, training set are rescaled from 768×768 to 256×256 , imposed with 20% zoom, 20% shear transformation, $1/255$ ratio of pixel values normalisation and random horizontal and vertical flip. Random flip means that not all of images are flipped, so there are four possibly direction pattern in actual data in training procedure. Meanwhile in "Whole

Automatic identification of ships on satellite images

DataSet” mode and test set pre-processing, images are only rescaled to 256×256.

Table 3-3 below demonstrates the how data augmentation works.

Table 3-3 Demonstration of Data Augmentation

| Pattern | Image | | | Description |
|---|-------|---|--|---|
| Original Image | |  | | Resolution: 768×768 |
| Test Set Training Set (“Whole DataSet”) | |  | | Resolution: 256×256 |
| Training Set (“Sampled”) (Pattern 1) | |  | | Resolution: 256×256 Horizontal flip: Disabled Vertical flip: Disabled |
| Training Set (“Sampled”) (Pattern 2) | |  | | Resolution: 256×256 Horizontal flip: Disabled Vertical flip: Enabled |
| Training Set (“Sampled”) (Pattern 3) | |  | | Resolution: 256×256 Horizontal flip: Enabled Vertical flip: Disabled |
| Training Set (“Sampled”) (Pattern 4) | |  | | Resolution: 256×256 Horizontal flip: Enabled Vertical flip: Enabled |

3.4.3 Package “Evaluation”

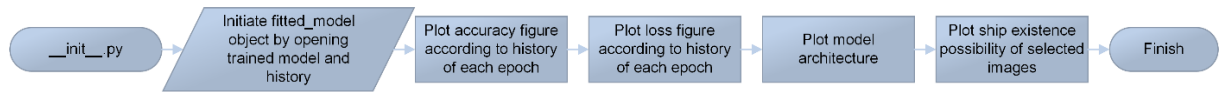


Figure 3.10 Flow chart of "Evaluation__init__.py"

Figure 3.10 shows the procedure of model evaluation. Actually, evaluation is mainly the visualisation of model training result. In this package, model accuracy, model loss, model architecture figures are generated using model history and model itself respectively. Finally, by calling trained model again, possibility of ship existence on some selected images are demonstrated in both pyplot figures and command line.

For the accuracy, loss and possibility result, please refer to “4.1 Results” section.

3.4.4 Package “ModelsTraining”

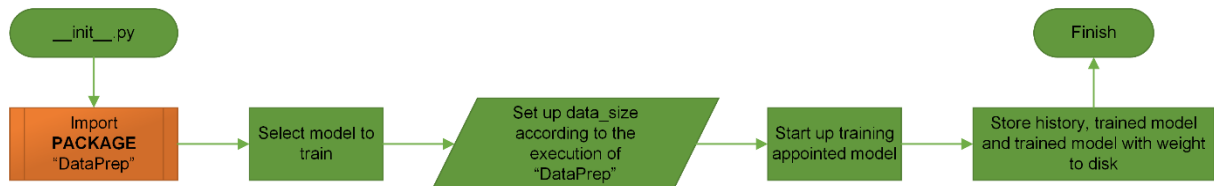


Figure 3.11 Flow chart of "ModelsTrainig__init__.py"

This package is the core area to this project. In Figure 3.11, module “__init__.py” first executes all the code in package “DataPrep”, then starts training selected deep learning model.

Rest of modules in this package are designed deep learning model written in Keras. Table 3-4 below provides important information of model trained in this project, including training approach, number of total and trainable parameters, postfix of names and remarks to each model.

Table 3-4 Information of Models Trained in this Project

| Model Name | Training Approach | #Total Parameters | #Trainable Parameters | Name Postfix |
|-------------------|-------------------|-------------------|-----------------------|--------------|
| AlexNet | Traditional | 58,285,441 | 58,285,441 | / |
| AlexNetMod_D | Traditional | 4,943,617 | 4,943,617 | Mod_D |
| AlexNetMod_D_F | Traditional | 552,673 | 552,673 | Mod_D_F |
| AlexNetMod_D_F_KS | Traditional | 7,597,921 | 7,597,921 | Mod_D_F_KS |
| CNNBenchmark | Traditional | 3,706,049 | 3,706,049 | / |
| ResNet50PT | Transfer | 158,857,089 | 135,269,377 | PT |
| VGG16Mod_D_F | Traditional | 6,180,945 | 6,180,945 | Mod_D_F |
| VGG16PT | Transfer | 33,601,345 | 18,886,657 | PT |
| VGG16 | Traditional | 134,264,641 | 134,264,641 | / |

Automatic identification of ships on satellite images

Four types of architectures are successfully implemented, three of them are prize-winning model that debut on Computer Vision and Pattern Recognition (CVPR) conference: ResNet, VGG and AlexNet (Krizhevsky, Sutskever, & Hinton, 2012). Model “CNNBenchmark” is inspired by a kernel published on Kaggle.com (Bezirganyan, 2018).

In order to explore the way to increase performance through architecture, modification on several layers of some models are tried out. These modified models are marked with name postfix “Mod”. Meanwhile, “D”, “F”, “KS” that placed right after “Mod” stand for model’s dense layer, filter size and kernel size/strides are modified respectively. What’s more, name postfix “PT” is abbreviation of “Pre-Trained”, indicates that transfer learning is the approach of training certain models.

For the architecture detail of each deep learning model, please refer to “Appendix II: Architecture Summary of Implemented Models” section.

3.4.5 Package “Utilities”

In this package, module “CSVOpen.py” defines a class to open certain CSV files and read them. It features exception handling to solve the problem when CSV file is missing or corrupted.

Module “DatasetDirectory.py” defines a class that can be instantiated to store dataset directory. Object derived from this class is callable by any module in this application

Module “SelectionValidation.py” defines a class that checks erroneous in user’s inputs throughout the entire application by comparing actual inputs with designed inputs.

Module “TimeStamp.py” defines a class that provides time stamp on start and stop of an operation. It also features calculation on time usage of certain operation.

3.4.6 Default Package

Module “StartUp.py” is the application main module that implements module “Menu.py” and records log file using command “tee”.

Module “Menu.py” allows user choosing different application functions to execute.

Module “GlobalVariable.py” provides reference to an object recording user-defined dataset directory to all packages and modules in this project, since dataset directory can be frequently used in nearly everywhere.

Chapter 4: Results and Discussion

4.1 Results

4.1.1 Outcomes from Application

Since the application includes several functions, files that generated may be varied according to the selected function. Table 4-1 below shows outcomes from each feature with corresponding descriptions.

In the table below, there are marks [Date], [ModelName] and [ImageId]. [Date] is the formatted date that application finished as “YYYY-MM-DD”, where year YYYY, month MM and day of month DD. [ModelName] is the model trained or evaluated during application execution e.g. “CNNBenchmark”. [ImageId] is name of image that used in ship existence prediction.

Table 4-1 Generated Files from Application

| Menu | Generated File | Description |
|-------------------------------------|--|---------------------------------|
| Analysis | TrainingSetAnalysis.csv | Dataset attribute |
| | TestSetAnalysis.csv | Dataset attribute |
| Model Training (Pre-processing) | Statistics of Appointed Dataset Scope.png | Dataset visualisation |
| | Statistics of Actual Training Set Data.png | Dataset visualisation |
| | Statistics of Actual Test Set Data.png | Dataset visualisation |
| | [Date]_Sample Sample.jpg train_config_*.jpg test_config_*.jpg | Data augmentation visualisation |
| Model Training (Actual Training) | [Date] [ModelName] trained_model.h5 | Fitted model |
| | [Date] [ModelName] trained_model_weights.h5 | Model weight |
| | [Date] [ModelName] history.txt | Training history |
| Evaluation | [Date]_[ModelName]_Accuracy.png | Accuracy visualisation |
| | [Date]_[ModelName]_Loss.png | Loss visualisation |
| | [Date]_[ModelName]_Summary.svg | Architecture visualisation |
| | [Date]_[ModelName]_Prediction_[ImageId].png | Prediction visualisation |
| All Features | [Date] log.txt | Application log |

4.1.2 Model Accuracy and Loss

Tables below show each model's accuracy and loss. Due to the variance on training batch size, performance figures are segmented into two tables: small batch size on Table 4-2 and large batch size on Table 4-3. Indices on both tables results from the last training epoch (i.e. 20th).

On following tables, "N/A" stands for certain deep learning model is not trained, performance result is not applicable. "OOM", as known as "Out of Memory", means the deep learning model cannot be trained due to limitation of device memory.

Table 4-2 Accuracy and Loss of Each Model (Small Batch Size: 2)

| Model | Loss (Training Set) | Accuracy (Training Set) | Loss (Test Set) | Accuracy (Test Set) |
|-------------------|--------------------------------|------------------------------------|----------------------------|--------------------------------|
| AlexNet | 5.2110 | 0.6767 | 5.6683 | 0.6483 |
| AlexNetMod_D | 0.6319 | 0.6735 | 0.6462 | 0.6521 |
| AlexNetMod_D_F | N/A | N/A | N/A | N/A |
| AlexNetMod_D_F_KS | 0.6318 | 0.6736 | 0.6529 | 0.6485 |
| CNNBenchmark | 0.3372 | 0.8743 | 0.3332 | 0.8780 |
| ResNet50PT | 5.1706 | 0.6792 | 5.6683 | 0.6483 |
| VGG16Mod_D_F | 0.2111 | 0.9226 | 0.2076 | 0.9228 |
| VGG16PT | 5.3174 | 0.6701 | 5.6392 | 0.6501 |
| VGG16 | 10.7499 | 0.3257 | 10.2594 | 0.3565 |

Table 4-3 Accuracy and Loss of Each Model (Large Batch Size: 100/250/1000)

| Model | Loss (Training) | Accuracy (Training) | Loss (Test) | Accuracy (Test) | Batch Size |
|-------------------|----------------------------|--------------------------------|------------------------|----------------------------|-----------------------|
| AlexNet | 0.2610 | 0.8952 | 0.2747 | 0.8891 | 250 |
| AlexNetMod_D | 0.2540 | 0.9008 | 0.2576 | 0.8958 | 250 |
| AlexNetMod_D | 0.2937 | 0.8795 | 0.2797 | 0.8820 | 1000 |
| AlexNetMod_D_F | 0.2296 | 0.9124 | 0.2557 | 0.8972 | 250 |
| AlexNetMod_D_F_KS | 0.1768 | 0.9341 | 0.1926 | 0.9317 | 250 |
| CNNBenchmark | 0.2112 | 0.9181 | 0.2511 | 0.9014 | 250 |
| ResNet50PT | 5.2252 | 0.6758 | 5.5897 | 0.6532 | 100 |
| ResNet50PT | 5.2767 | 0.6726 | 5.5897 | 0.6532 | 250 |
| VGG16Mod_D_F | 0.6141 | 0.6732 | 0.6200 | 0.6532 | 250 |
| VGG16PT | 0.2452 | 0.9075 | 0.2242 | 0.9152 | 100 |
| VGG16 | OOM | OOM | OOM | OOM | 100 |
| VGG16 | OOM | OOM | OOM | OOM | 250 |

4.1.3 Time Consumption of Model Training

Table 4-4 shows training time of each model trained by different batch size.

Table 4-4 Training Time of Models on (second)

| Model | Small Batch Size = 2 | Large Batch Size | #Batch Size |
|-------------------|----------------------|------------------|-------------|
| AlexNet | 29616.53125 | 8879.78125 | 250 |
| AlexNetMod_D | 23853.53125 | 8939.703125 | 250 |
| | | 9122.53125 | 1000 |
| AlexNetMod_D_F | N/A | 8756.59375 | 250 |
| AlexNetMod_D_F_KS | 23967.28125 | 9188.78125 | 250 |
| CNNBenchmark | 22478.203125 | 9175.546875 | 250 |
| ResNet50PT | 39091.6875 | 10414.4375 | 100 |
| | | 10176.125 | 250 |
| VGG16Mod_D_F | 24896.0 | 9461.4375 | 250 |
| VGG16PT | 26638.0 | 11202.546875 | 100 |
| VGG16 | 31185.359375 | OOM | 100 |
| | | OOM | 250 |

4.1.4 Model Performance Visualisation

The figures on following pages are the visualisation of accuracy and loss value calculated using training set and test set on each training epoch. Among multiple results of each trained model, only the one with the highest accuracy and lowest loss value are visualised in this report.

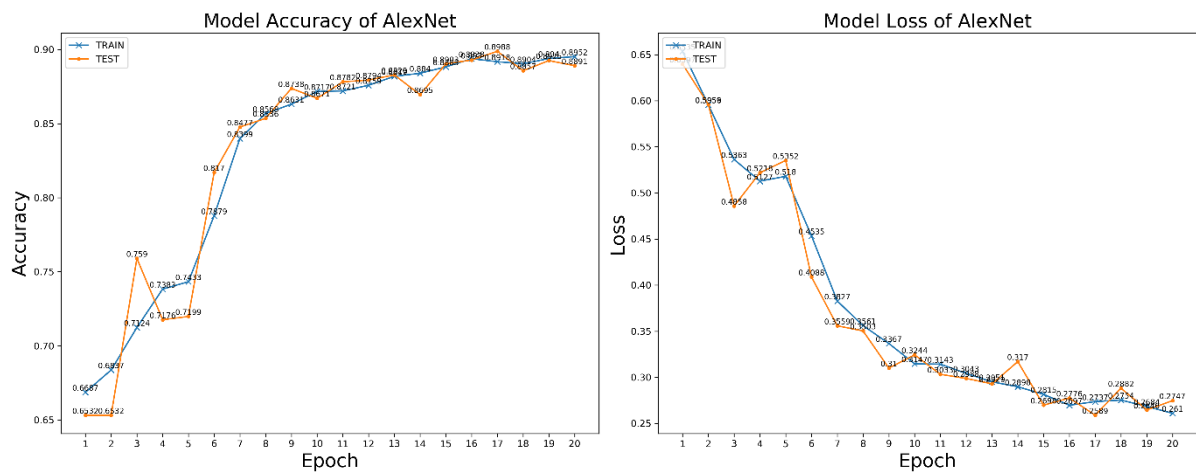


Figure 4.1 Model Accuracy and Loss of original “AlexNet” (Batch Size = 250)

Automatic identification of ships on satellite images

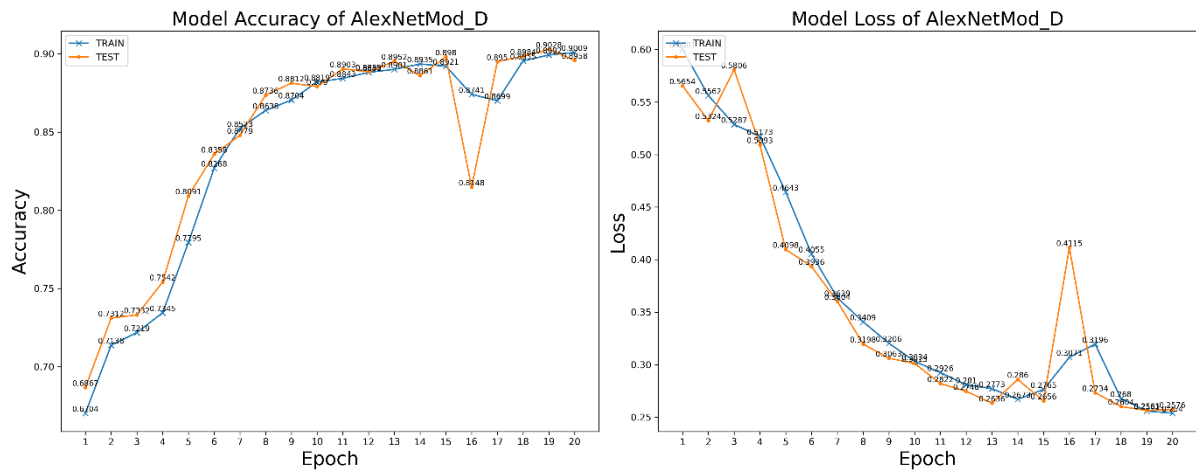


Figure 4.2 Model Accuracy and Loss of “AlexNetMod_D” (Batch Size = 250)

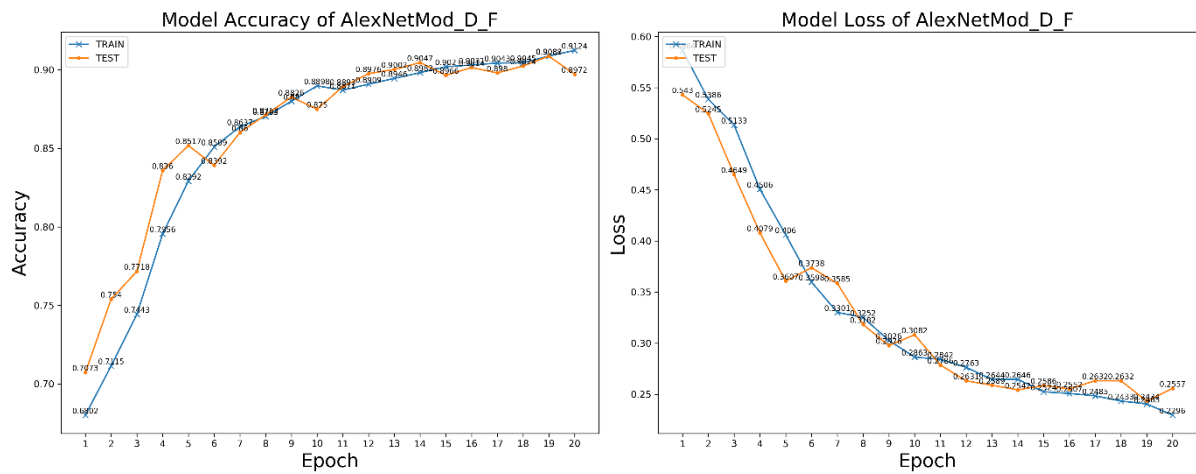


Figure 4.3 Model Accuracy and Loss of “AlexNetMod_D_F” (Batch Size = 250)

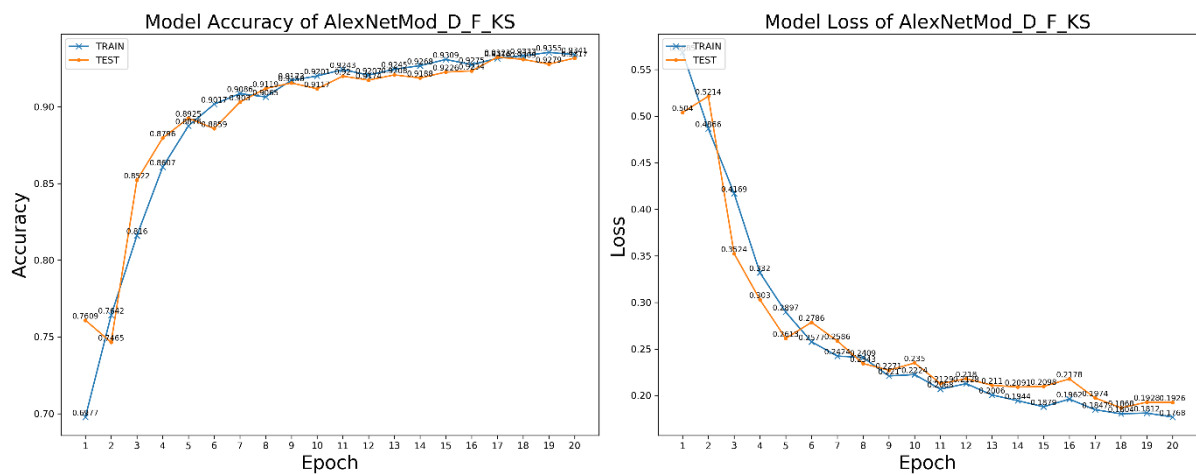


Figure 4.4 Model Accuracy and Loss of “AlexNetMod_D_F_KS” (Batch Size = 250)

Automatic identification of ships on satellite images

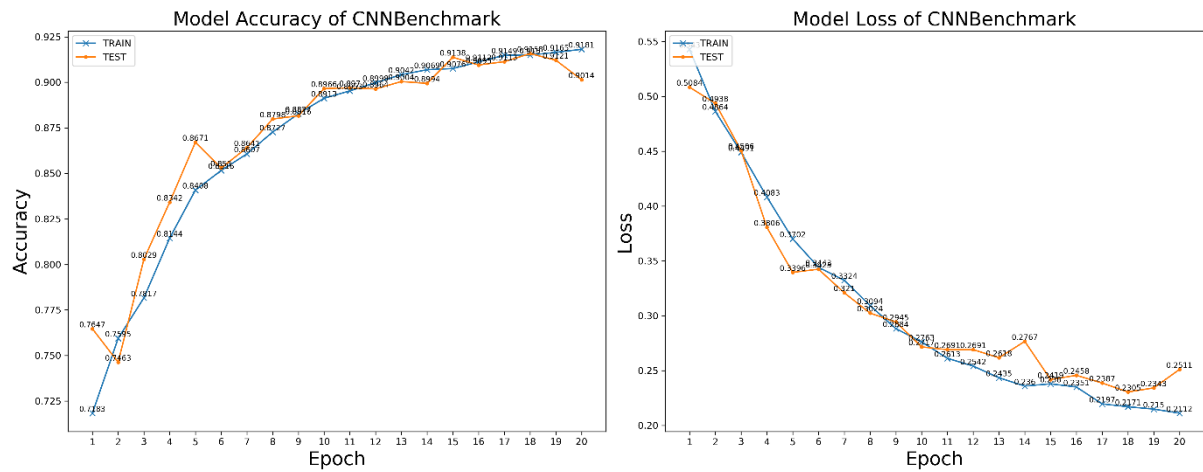


Figure 4.5 Model Accuracy and Loss of “CNNBenchmark” (Batch Size = 250)

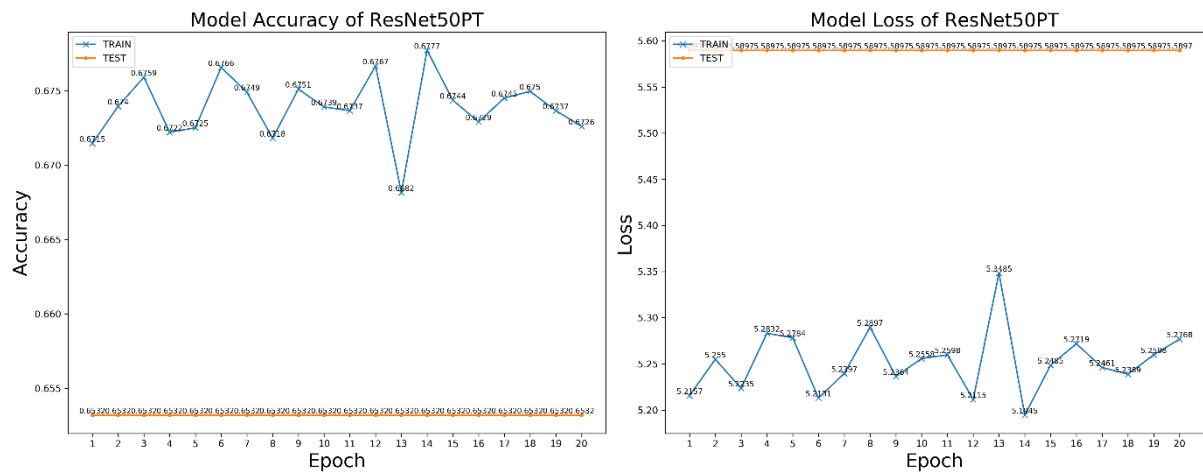


Figure 4.6 Model Accuracy and Loss of Finely Tuned “ResNet50PT” (Batch Size = 250)

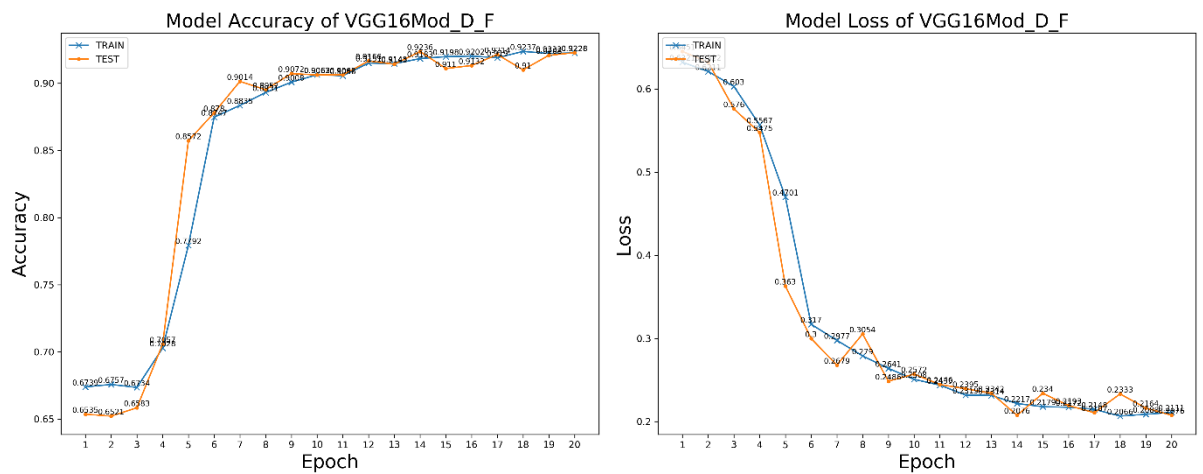


Figure 4.7 Model Accuracy and Loss of “VGG16Mod_D_F” (Batch Size = 2)

Automatic identification of ships on satellite images

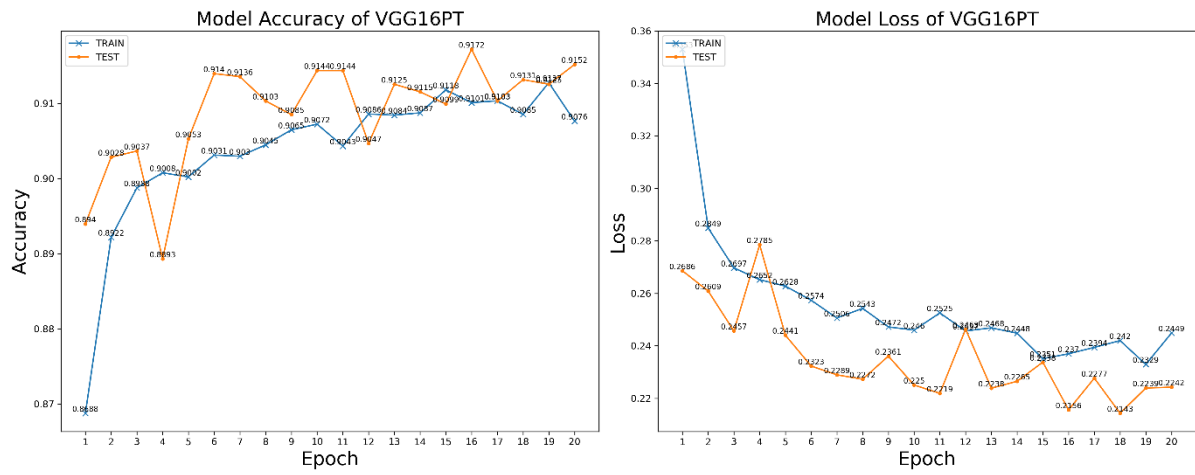


Figure 4.8 Model Accuracy and Loss of Finely Tuned “VGG16PT” (Batch Size = 100)

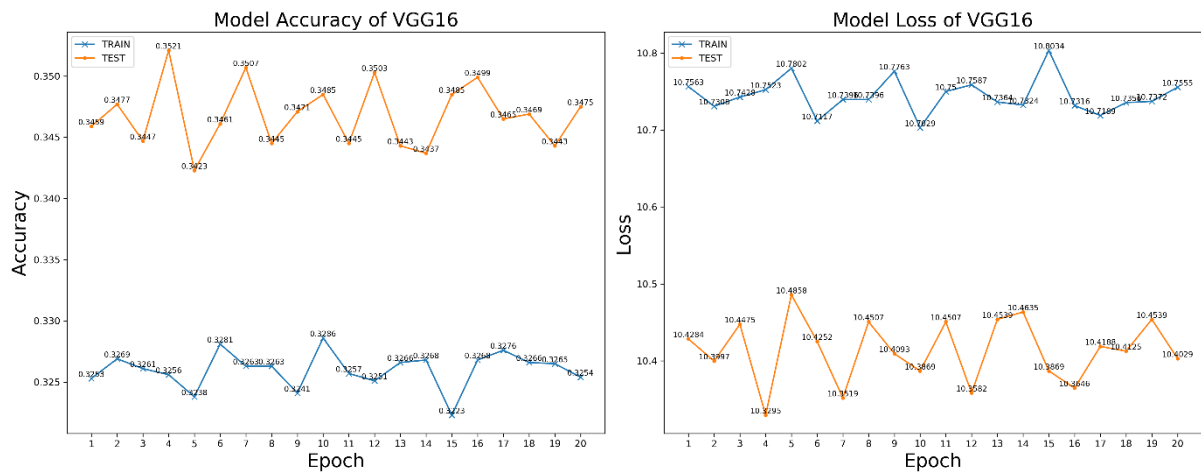


Figure 4.9 Model Accuracy and Loss of original “VGG16” (Batch Size = 2)

4.1.5 Ship Existence Prediction Visualisation

In addition to visualisation on performance indices, prediction on images about ship existence is also demonstrated concretely. From Figure 4.10 it can be observed that four images, randomly chosen from provided test set and rescaled to 256×256 , are inputted into trained model “CNNBenchmark”. Possibilities revealing ship existence are attached on examined images.

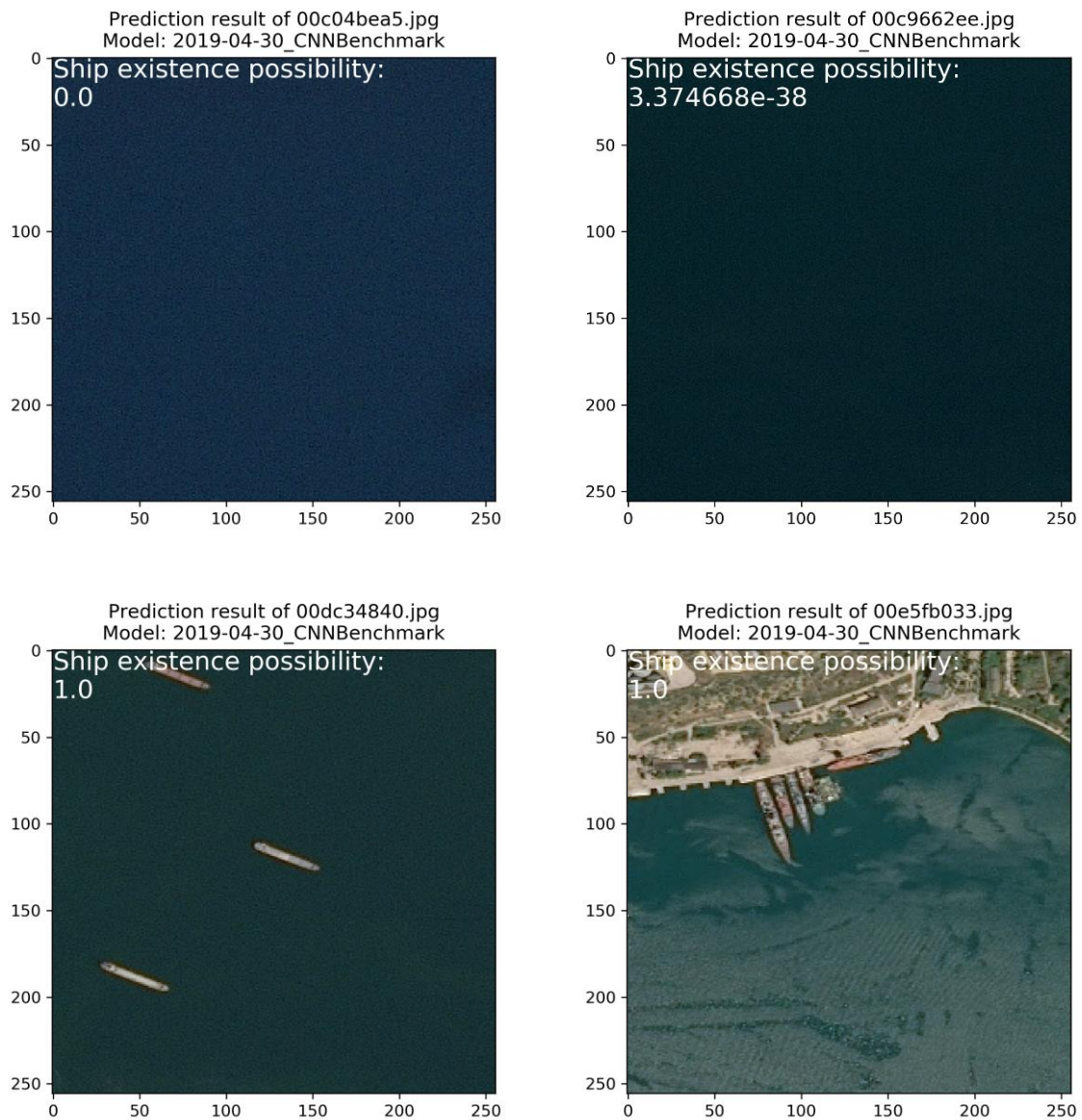


Figure 4.10 Ship Existence Prediction Visualisation

Chapter 5: Conclusion and Further Work

5.1 Conclusion about Outcomes

In this project, an application that can explore dataset attribute, pre-process dataset images, train deep learning models and evaluate model performance is implemented. Nine kinds of deep learning models in total are tried out in this project, and seven of them have achieved around 90% accuracy. What's more, visualisation on these aspects are successfully realised:

- Statistical data about ship existence of labelled dataset (“3.4.2 Package “DataPrep””);
- Data augmentation configuration (“3.4.2 Package “DataPrep””);
- Model performance indices of accuracy and loss (“4.1.4 Model Performance Visualisation”);
- Prediction of ship existence on selected images (“4.1.5 Ship Existence Prediction Visualisation”);
- Model architecture (“Appendix II: Architecture Summary of Implemented Models”);

5.2 Conclusion about Model Performance and Training Time

5.2.1 Model with the Best Performance

According to table Table 4-2 and Table 4-3, the model with the best performance is “AlexNetMod_D_F_KS” which reaches 93% accuracy in test set. This model is trained in 20 epochs and batch sized 250.

5.2.2 Factors that Impact on Performance

All following conclusions are derived from table Table 4-2 and Table 4-3.

- **Depth of model architecture**

Increase on model architecture depth is not proportional to model accuracy improvement.

For instance, in group of training batch size 250, accuracy of “VGG16Mod_D_F” with deeper network architecture is not better than that of “AlexNet” and its modified versions. To solve binary classification problems like ship detection, choosing a somewhat shallow CNN is beneficial to get ideal result.

- **Configuration of model architecture (i.e. number of filters, kernel size, stride)**

Cropping the number of filters can improve the accuracy.

This can be observed from the comparison between “AlexNetMod_D” and “AlexNetMod_D_F” trained with batch size 250. The reason is that number of filters is equivalent to the feature fragmentation. Since shape of ships is not complex and can be fragmented into limited pieces, cutting down the number of filters can improve the accuracy.

Selecting smaller kernel size and stride can help the computer collect more feature information, thus enhance the accuracy, because kernel size and stride are determination to the degree of pixel information collecting.

It can be known from the tables that accuracy of “AlexNetMod_D_F_KS” is higher than “AlexNetMod_D_F” trained with batch size 250. However, shrinking the kernel size and stride has the risk of expanding the number of trainable parameters, which may cause OOM in some devices.

- **Batch size of training**

As for shallow CNN like “CNNBenchmark” and “AlexNet” with its modifications, expanding training batch size leads to a giant leap in accuracy. However, for deeper CNN like “VGG16” family and “ResNet50”, expanding training batch size is not very effective on elevating accuracy.

5.2.3 Factors that Impact on Training Time

All following conclusions are derived from Table 4-4.

- **Scale of dataset**

Scale is proportional to time cost in training.

Dataset scale has the most significant impact on training time since operation of model training is based on images. Since every model in this project are trained using 25,000 images, training time of model in the same batch size group is relatively close, especially for the models trained using large batch size.

- **Number of trainable parameters (model architecture)**

There is no evident relationship between number of trainable parameters (model architecture) and training time. Difference in architecture will lead to difference on training time.

- **Batch size of training**

Selecting relatively large batch size can accelerate model training.

Batch size is another vital factor that has influence on training time. According to the table, training time under small batch size is longer than training time under large batch size. However, by examining the table, it can be also seen that in model “AlexNetMod_D”, training under batch size 1000 is slower than that under batch size 250. Therefore, relatively large, not too large batch size selection can accelerate model training.

5.3 Problems and Solutions

5.3.1 Configuration on Software Drivers

Configuration on software drivers is the largest problem by mid-term. After setting up the environment in project early-term, I tried to run a sample program. Unfortunately, exception “Failed to load the native TensorFlow runtime.” and “ImportError: DLL load failed: The specified module could not be found” popped up in the CLI. By searching Internet, I found that two modules are ignored in setting up environment: Microsoft Visual C++ 2017 Redistributable and cuDNN (NVIDIA CUDA Deep Neural Network library). However, problem remains after adding up certain modules. Once again, I checked the TensorFlow requirement and forums, from which I found that the up-to-date version of Python 3.7.2 and CUDA driver 10.0 are not supported by TensorFlow 1.12. After downgrading components to the correct version (Python 3.6.8, CUDA 9.0, cuDNN 7.4.1.5), the problem is solved.

5.3.2 Model Accuracy Improvement

When training unmodified model “VGG16”, it can be found in Table 4-2 that its accuracy is much lower than that of “CNNBenchmark”, which is 30% to 87%. After discussed with my supervisor, I tried to lower the number of parameters by reducing filter number, adjusting kernel size and stride, but the accuracy remained unsatisfying. By reporting this issue with my supervisor and re-examining the model architecture once again, I found that the selection of activation function on final dense layer is incorrect. Since ship detection is a binary classification problem, it is proper to use activation function like “sigmoid” or “ReLU” which outputs nearly binary result, rather than to use “softmax” that is widely used in solving multiple classification problems. Eventually, the modified model “VGG16Mod_D_F” has reached recognition accuracy of 92%.

5.4 Project Redo Differences

If there is a chance to redo this project, more aspects that concerning on performance will be discussed. For instance, impacts of training epoch on performance will be explored. Plus, whole dataset will be used in models training to find impacts of dataset scale on performance. More importantly, other complicated CNN models will be tried to be implemented.

5.5 Further Work

5.5.1 Ship Type Recognition

Due to the characteristic of complex deep learning methods like “ResNet50” and “VGG16”, which are initially designed to recognize a huge number of objects, its architecture can be easily modified to recognize different kinds of ships. By collecting more ship satellite image, collaborating with database from professional ship manufactories and using acceptable labelling approach, the model is expected to distinguish different types of ships. Once the system can recognise a variety of ship types, the application scenario would be broadened: the system can keep tracking specific ships and then predict their behaviours.

5.5.2 Improve Accuracy by Combining Multiple Remote Sensing

Airbus dataset used in this project can be classified as “Visible Spectral Remote Sensing”. In fact, in remote sensing area, there are also other remote sensing approaches apart from the one on visible spectral. For example, various satellites are equipped with laser camera, infrared camera or Synthetic Aperture Radar (SAR). These kinds of equipment dramatically expanded the scope that human could sense, since they are able to provide more information from multiple dimensions. It is believed that by combining multiple remote sensing methods, the accuracy of ship detecting will be enhanced further, because the combination could polarise the result: it increases the possibility of determined results and meanwhile eliminates the fluctuation and incorrect results.

5.5.3 Advanced User-friendly Engineering

To make the application genuinely meet the need of cargo companies, some engineering on user-friendly issues should be tackled. For instance, a well-designed graphic user interface (GUI) may provide a more intuitive interaction approach with users. GUI is powerful especially in inputting operations. Furthermore, some operation like dataset analysis could be accelerated to reduce time cost.

References

- Bezirganyan, G. (2018, Aug. 15). *Detect if there is a ship by CNN (base model)*. Retrieved Feb. 11, 2019, from <https://www.kaggle.com/grigorbezirganyan/detect-if-there-is-a-ship-by-cnn-base-model>
- Brownlee, J. (2017, Dec. 20th). *A Gentle Introduction to Transfer Learning for Deep Learning*. Retrieved from machinelearningmastery.com: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>
- Hargrave, M. (2019, Mar. 14th). *Deep Learning Definition*. Retrieved March 20th, 2019, from investopedia.com: <https://www.investopedia.com/terms/d/deep-learning.asp>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV*, pp. 770-778.
- Krizhevsky, A., Sutskever, I., & Hinton, G. H. (2012). ImageNet Classification With Deep Convolutional Neural Networks. *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1097-1105.
- LeCun, Y., Bottou, L., Yoshua, & Haffner, P. (1998, Nov.). Gradient-Based Learning Applied to Document. *PROC. OF THE IEEE*, pp. 1-46.
- ML Cheatsheet. (2017). *Gradient Descent*. Retrieved from readthedocs.io: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html
- Ng, A. (2018). *Neural Networks and Deep Learning | Coursera*. Retrieved from Coursera.org: <https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning>
- Rouse, M. (2018, April). *DEFINITION - Convolutional Neural Network*. Retrieved March 20th., 2019, from <https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network>
- Simonyan, K., & Zisserman, A. (2014, September 4th). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.
- Stanford University. (2017). *CS231n Convolutional Neural Networks for Visual Recognition*. Retrieved from cs231n.github.io: <http://cs231n.github.io/convolutional-networks/#conv>

Acknowledgement

To begin with, I would like to express appreciation to my supervisor who provided and supported this project title. During the development of this project, he offered me a lot of help and instructions on both project and paperwork. What have left me deep impression are that he has given me some hint to improve the accuracy of model training as well as the overall principle of finishing a project, which is push forward the progress as fast as possible with reasonable planning. His patience and expertise have greatly assisted me to step over obstacles and eventually finish the project.

I am also grateful to Kaggle.com and Airbus for providing dataset used in this project for free. Furthermore, I would like to thank to Keras organisation and Google TensorFlow team who make the development of machine learning much easier and more intuitive.

In addition, I am thankful to all the online courses about deep learning like *Deep Learning Specialised* from deeplearning.ai and CS231n from Stanford University. These courses give me a clear view about deep learning mechanism.

What's more, I am grateful to my internship boss Kaku Chin Ph.D. who graduated from University of Tokyo. He helped me broaden up my mind on finishing this project and made me clearer about research direction. He also introduced some of his research experience to me, which let me realise the significance of time planning, thus let me develop the project smoother.

Appendix

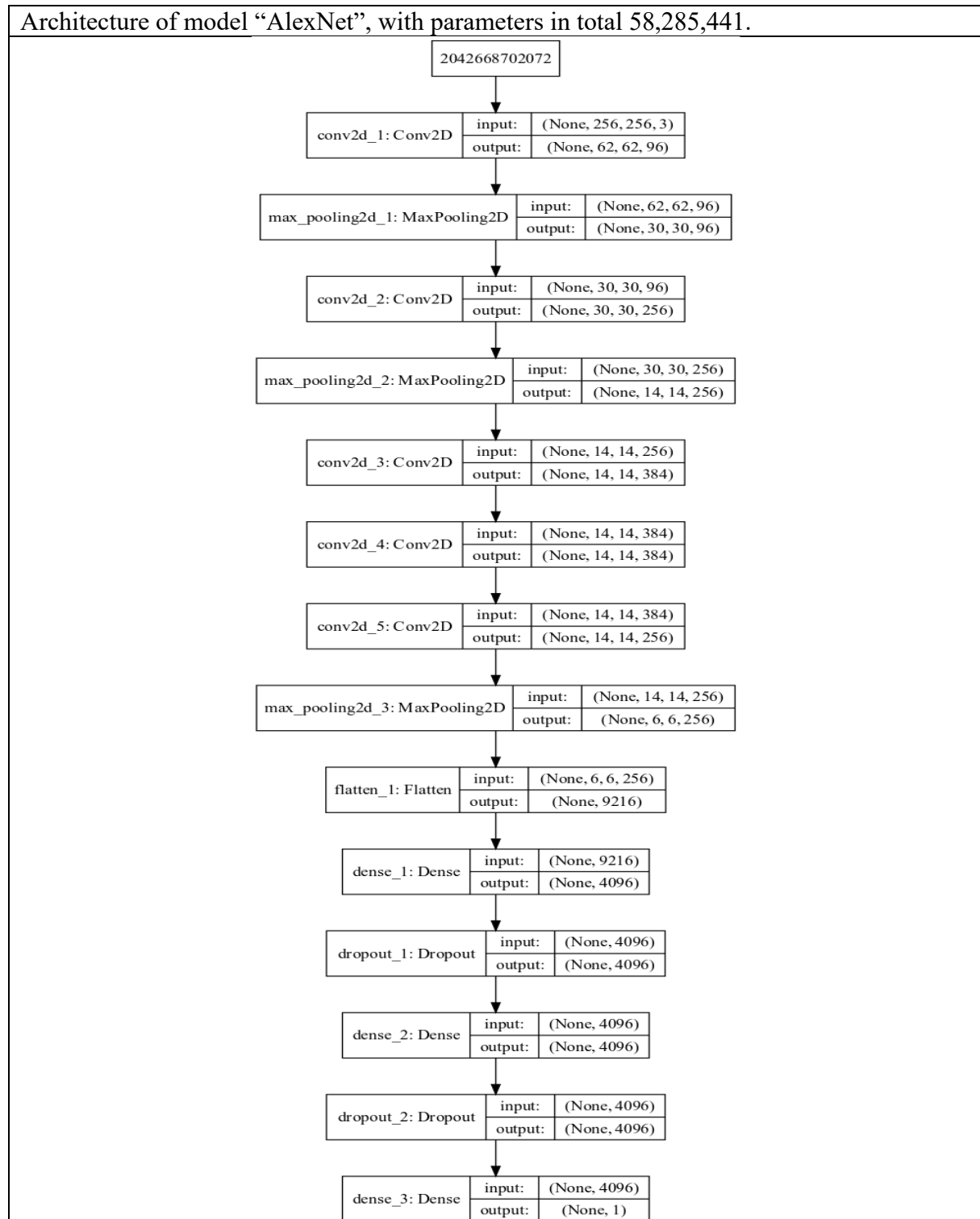
I Python Environment Package Dependencies

| Package Name | Version |
|---------------------|----------|
| Keras | 2.2.4 |
| Keras-Applications | 1.0.7 |
| Keras-Preprocessing | 1.0.9 |
| Markdown | 3.0.1 |
| Pillow | 5.4.1 |
| Pillow-PIL | 0.1.dev0 |
| PyWavelets | 1.0.2 |
| PyYAML | 5.1 |
| Werkzeug | 0.15.1 |
| absl-py | 0.7.1 |
| astor | 0.7.1 |
| cloudpickle | 0.8.1 |
| cycler | 0.10.0 |
| dask | 1.1.4 |
| decorator | 4.4.0 |
| gast | 0.2.2 |
| graphviz | 0.10.1 |
| grpcio | 1.19.0 |
| h5py | 2.9.0 |
| kiwisolver | 1.0.1 |
| matplotlib | 3.0.3 |
| networkx | 2.2 |
| numpy | 1.16.2 |
| pandas | 0.24.2 |
| pip | 19.0.3 |
| protobuf | 3.7.0 |
| pydot | 1.4.1 |
| pydot-ng | 2.0.0 |
| pyparsing | 2.3.1 |
| python-dateutil | 2.8.0 |
| pytz | 2018.9 |
| scikit-image | 0.14.2 |
| scikit-learn | 0.20.3 |
| scipy | 1.2.1 |
| setuptools | 40.8.0 |
| six | 1.12.0 |
| tensorboard | 1.12.2 |
| tensorflow-gpu | 1.12.0 |
| termcolor | 1.1.0 |
| toolz | 0.9.0 |
| tqdm | 4.31.1 |
| wheel | 0.33.1 |

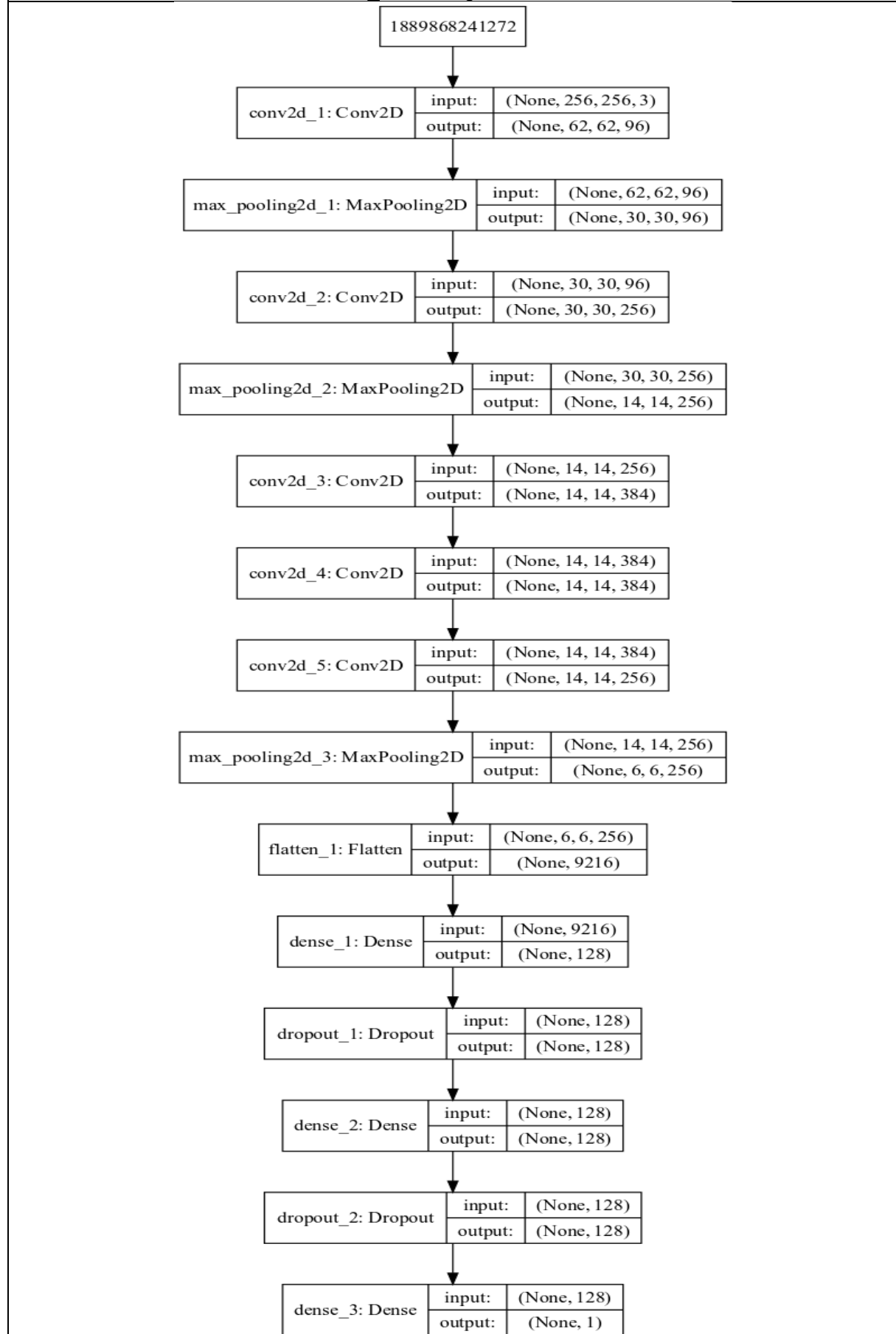
II Architecture Summary of Implemented Models

This part of appendix is the supplementary material to “3.4.4 Package “ModelsTraining”” in this report. The following graphs in a big table visualises the architecture of each implemented model in this project.

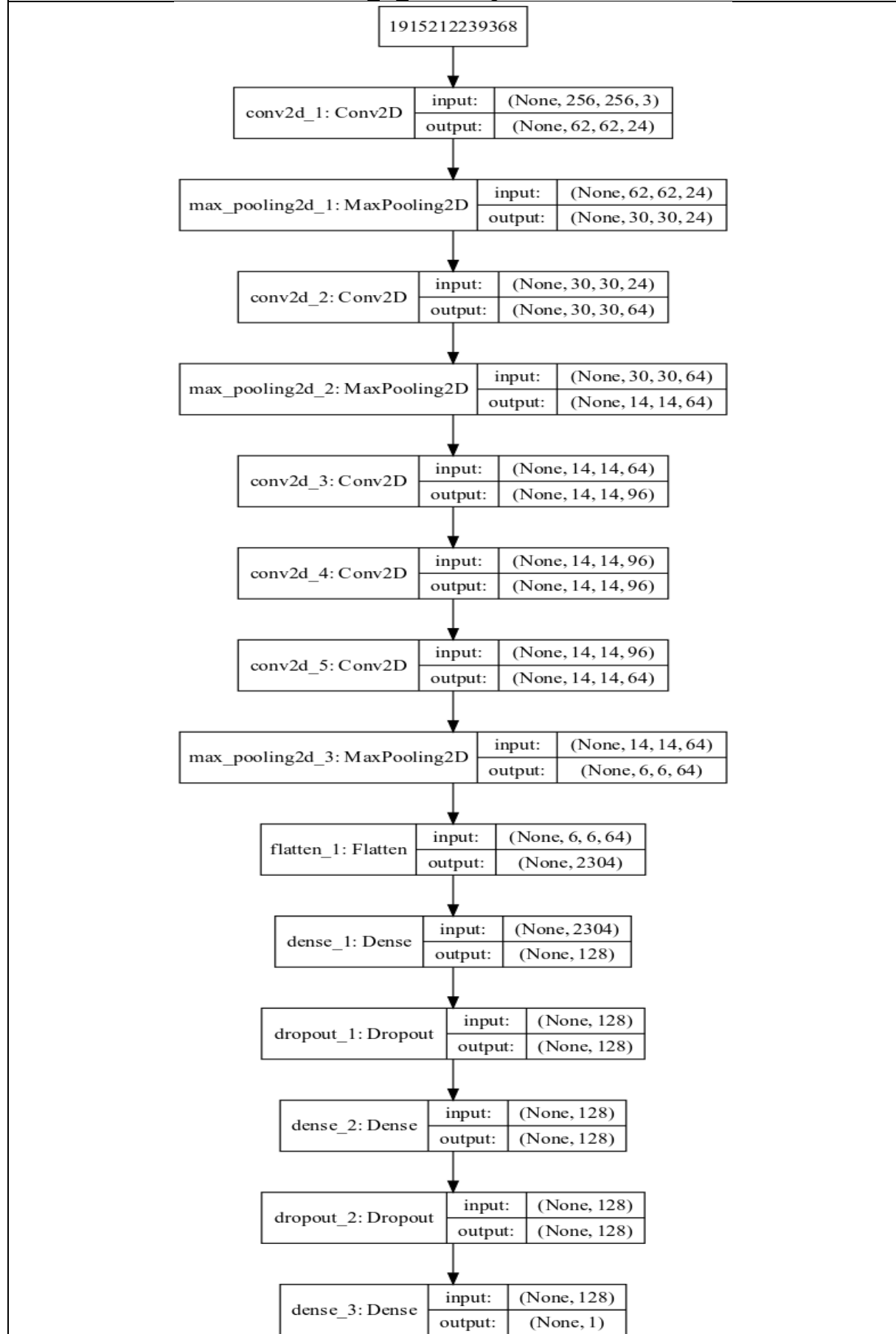
Table 5-1 Architecture Visualisation of Implemented Models



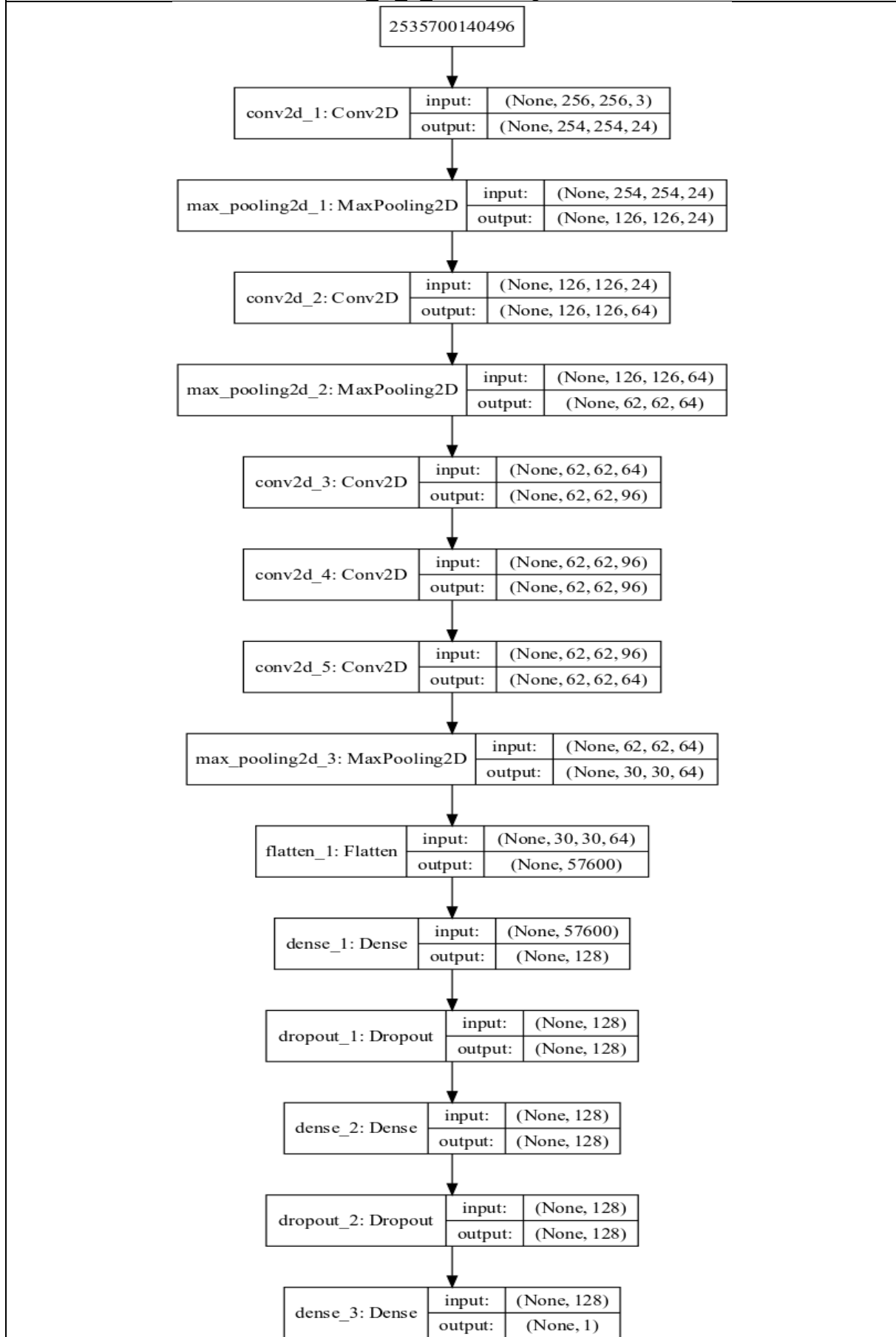
Architecture of model “AlexNetMod_D”, with parameters in total 4,943,617.



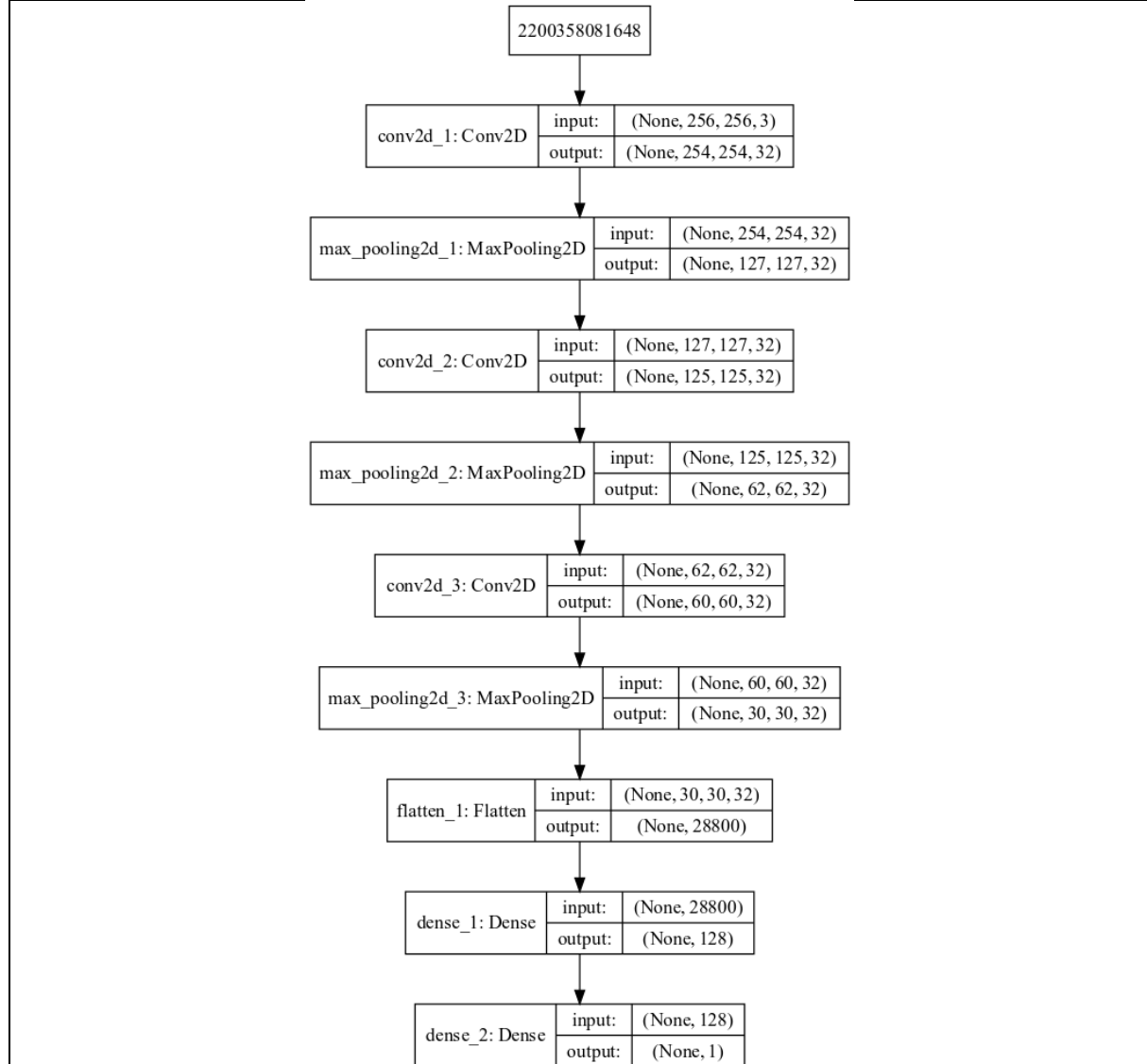
Architecture of model “AlexNetMod_D_F”, with parameters in total 552,673.



Architecture of model “AlexNetMod_D_F_KS”, with parameters in total 7,597,921.

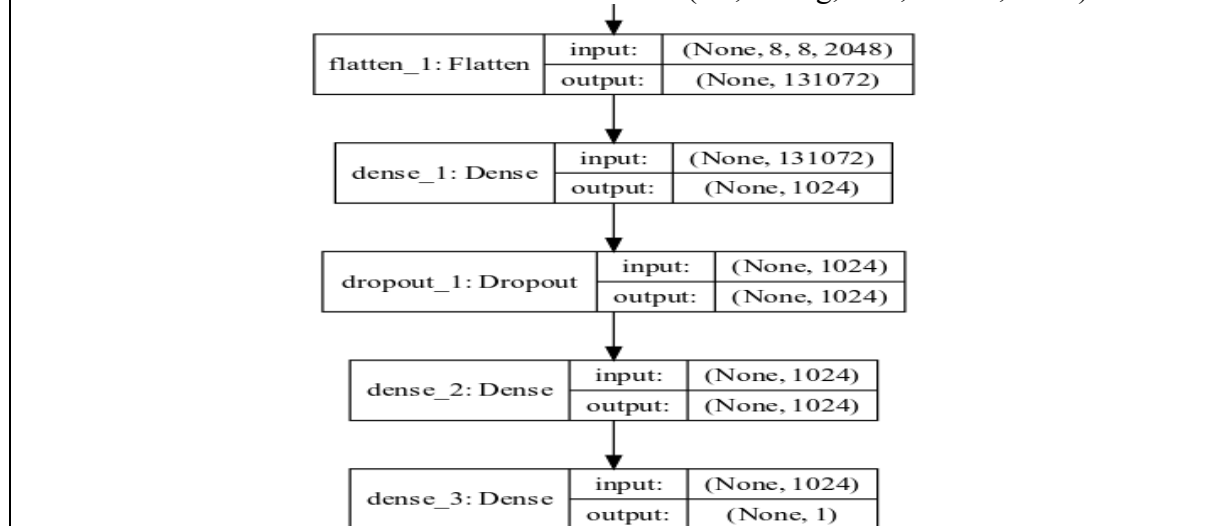


Architecture of model “CNNBenchmark”, with parameters in total 3,706,049.

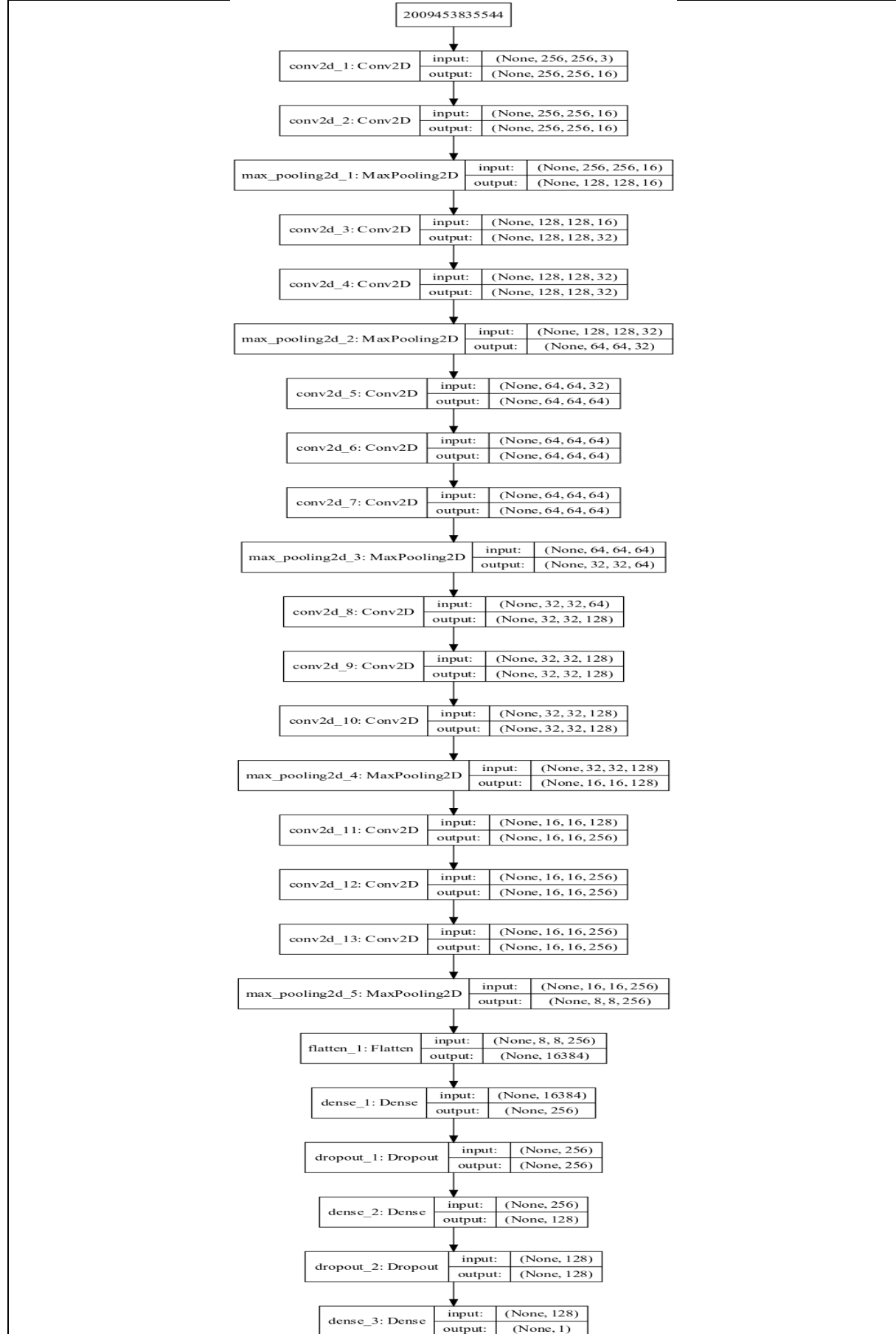


Architecture of model “ResNet50PT”, with parameters in total 158,857,089.

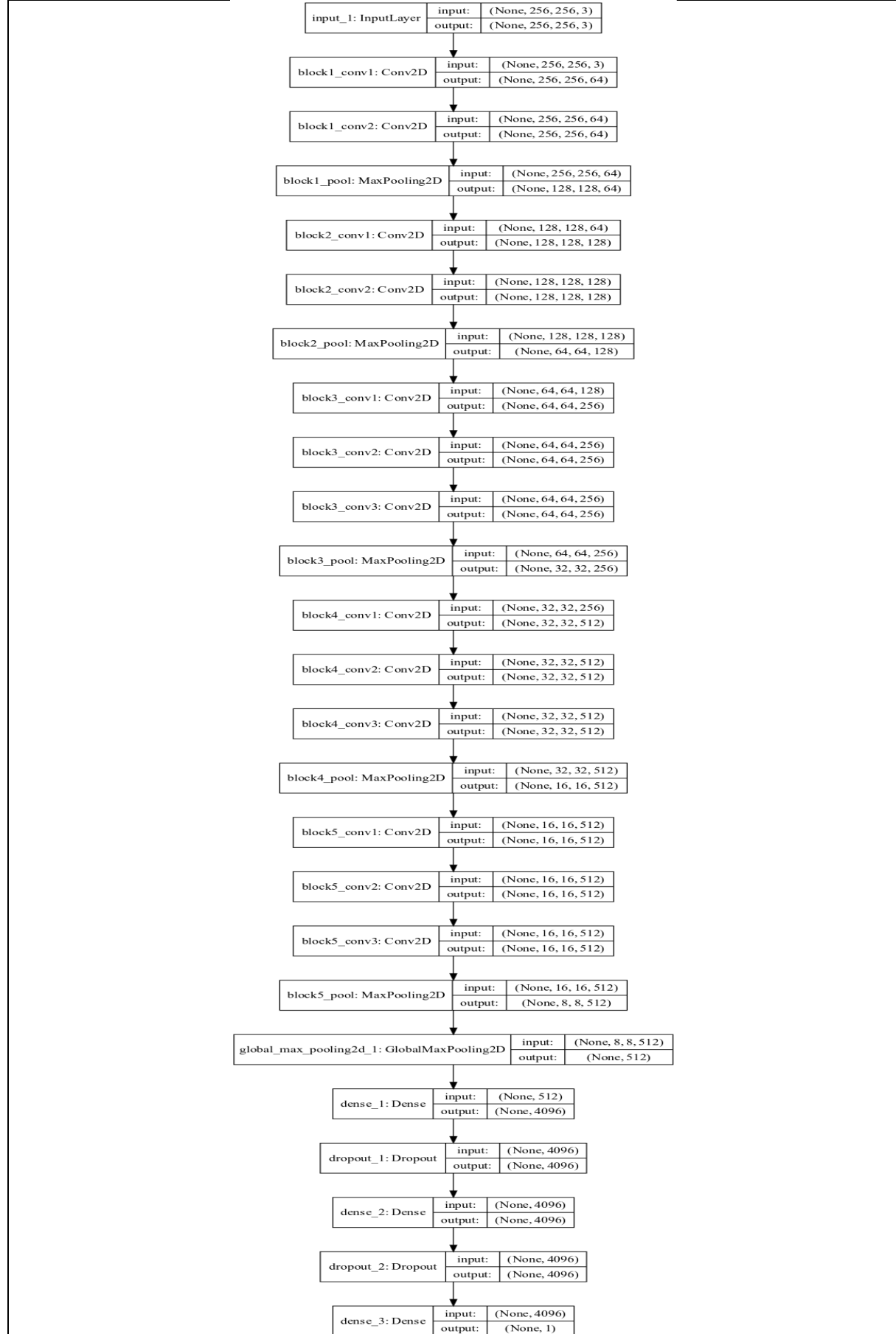
Since ResNet50 network are implemented in transferring learning approach, only fully connected layers at the end of this model are modified whereas main body of this model remains the same as its debut on CVPR conference (He, Zhang, Ren, & Sun, 2016).



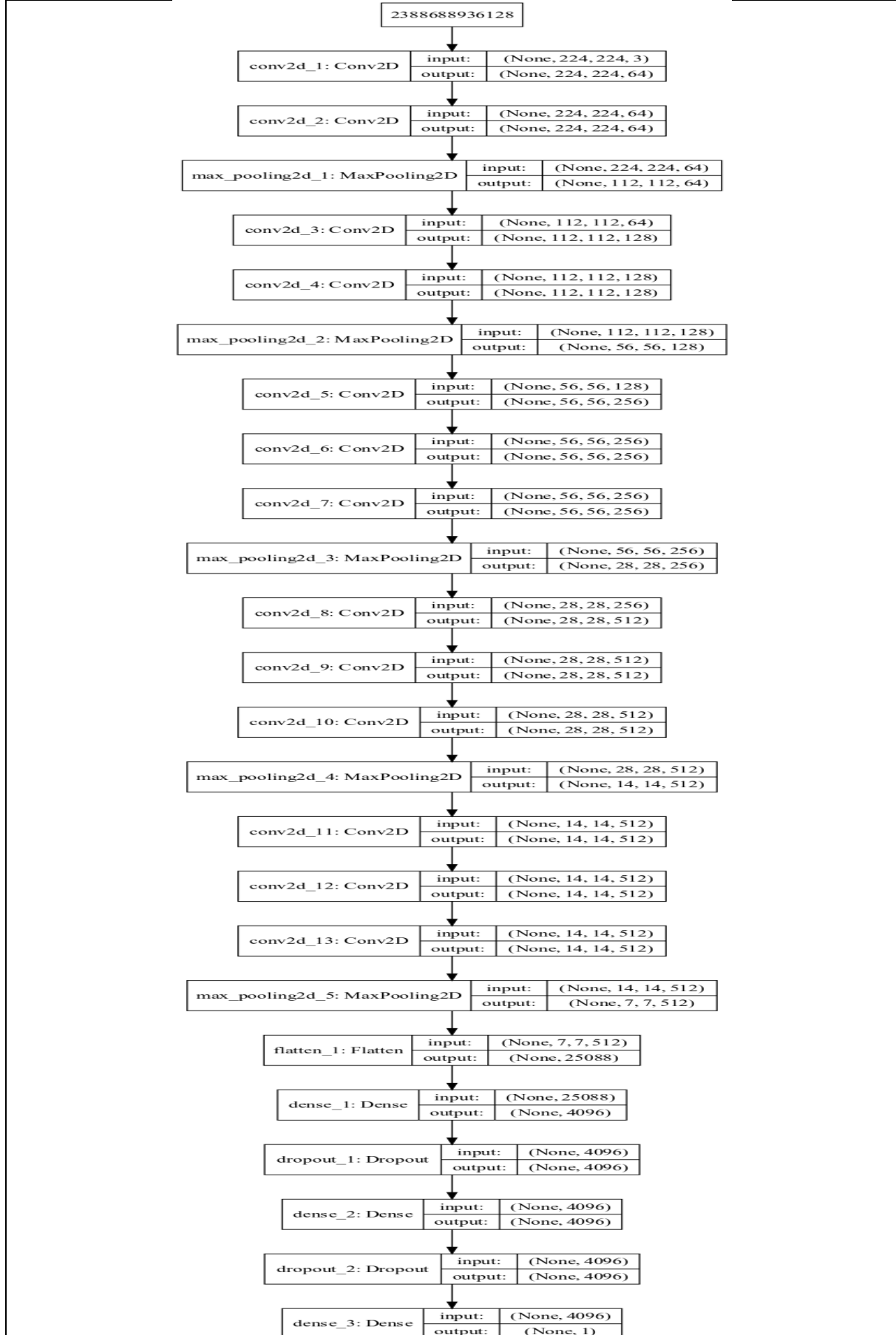
Architecture of model “VGG16Mod_D_F”, with parameters in total 6,180,945.



Architecture of model “VGG16PT”, with parameters in total 33,601,345.



Architecture of model “VGG16”, with parameters in total 134,264,641.



Risk Assessment

This project is based on software development, which does not possess any risks that prevent successful completion.

Automatic identification of ships on satellite images

Environmental Impact Assessment

This project is based on software development, which will not cause any damage to the environment.

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 1 – Supervisor

| | | | |
|-----------------------------|--|----------|----------|
| 论文题目 Project Title | Automatic identification of ships on satellite images | | |
| 题目分类 Scope | Information and Data Science | Research | Software |
| 主要内容 Project description | Nowadays, there is a pressing need to increase our ability to monitor shipping traffic, as shipping traffic has been growing quickly and so are the chances of problems ranging from environmental disasters to illegal activities. Satellite imaging provide a wealth of source of data that can be used for ship monitoring. However, the information contained in satellite images has not been fully exploted. In this project, state of the art data science approaches such as deep learning will be used to develop a method capable of correctly identifying ships from satellite images. A dataset from Airbus will be used to develop the data science method. | | |
| 关键词 Keywords | Image processing, machine learning, deep learning | | |
| 主要任务 Main tasks | 1 Learn relevant background on image analysis, machine learning and deep learning | | |
| | 2 Develop a data science method for ship detection | | |
| | 3 Evaluate the performance of the method and compare it with other methods | | |
| | 4 Compare the proposed olution with other methods | | |
| 主要成果 Measurable outcomes | 1 Software for visualising the chosen dataset | | |
| | 2 Software implementation of the developed method | | |
| | 3 Analysis of the performance of the proposed method | | |

北京邮电大学 本科毕业设计（论文）任务书

Project Specification Form

Part 2 - Student

| | | | | | |
|--|---|--------------------|--------------------------------|-------------|------------|
| 学院 School | International School | 专业 Programme | Internet of Things Engineering | | |
| 姓 Family name | 陈 Chen | 名 First Name | 兆雄 Zhaoxiong | | |
| BUPT 学号 BUPT number | 2015213445 | QM 学号 QM number | 151007198 | 班级 Class | 2015215121 |
| 论文题目 Project Title | 卫星图片中船舶的自动识别 Automatic identification of ships on satellite images | | | | |
| 论文概述 Project outline Write about 500-800 words Please refer to Project Student Handbook section 3.2 | <p>Project Background and Requirement Analysis: As the consequence of the development in economy and commercial, it can be witnessed a dramatic growth on the international shipping traffic. Indeed, the growth brings lots of revenue to the cargo industries. But it also raises series of challenges. The reason is that increased number of ships may lead to a burgeon of infraction manner at public sea. For instance, ship accidents that causing environmental contamination, piracy in Gulf of Aden and illegal fishing of whales. These matters damaged the regular transportation order, hence make people diffident to the shipping, causing a pessimistic prospect on cargo industry. Admittedly, more and more satellite pictures are collected to monitor the public sea. However, the data are not fully exploited due to the ignorance of developing more scientific analysing method, hence much more information was buried among the data sea.</p> <p>By analysing needs of customers in associated with background information, several key issue should be tackled:</p> <ol style="list-style-type: none"> Find a method to identify ships from current satellite image in more intuitive, convenient and fast-responsive characteristic. Help cargo enterprises anticipate threats and broadcasting alerts. (Eventually) Generate statistic data from satellite photos to help optimize the traffic schedule and improve the efficiency at sea. This term will need the help of real-time dataset, which is not applicable in this project. <p>Methodology: The project will be carried out in following procedures:</p> <ol style="list-style-type: none"> Early Stage <ol style="list-style-type: none"> Having a research and learning some knowledge about deep learning. All these materials could be retrieved from books, research papers, community forum and Coursera.com. The major aspects are: <ol style="list-style-type: none"> Deep learning algorithms. e.g. Convoluted Neural Network (CNN). Deep learning frames. e.g. TensorFlow/Caffe. Programming language. e.g. Python, JavaScript, C++ etc. Necessary software libraries. e.g. OpenCV for data pre-possessing and augmentation. Other matters not mentioned, which concerns about deep learning. Determining the algorithm and frame of prototype and the final version. | | | | |

| | |
|------------------------------|---|
| | <ol style="list-style-type: none"> 1.3. Building up dataset for this project. The dataset is provided by kaggle.com^[1], an online community of data scientists and machine learners owned by Google, Inc. in associated with Airbus S.A.S., a state-of-the-art aircraft manufacture. Apart from these datasets, no more data will be collected from other source for this project. 1.4. Preparing the dependencies on computer. One or two PCs with Intel processor and Nvidia GPU will be assigned. Also, the hyper computer from QMUL will be considered to speed up training section. 2. Main Stage <ol style="list-style-type: none"> 2.1. Developing the system. Agile method will be followed, which encourages programmer to develop the project from a prototype in a gradual, iterative way. A wide range of alteration will be allowed, according to the Agile Manifesto. The conceived iteration plan will be: <ol style="list-style-type: none"> 2.1.1. Dividing dataset into several groups, including training set, validation set and testing set. 2.1.2. Establishing certain algorithm on selected frame. 2.1.3. Code debugging. 2.1.4. Using online judgment system at kaggle.com to evaluate the performance of certain algorithm. 2.1.5. Adjusting the system based on the performance feedback. 2.1.6. Dataset augmentation. 2.2. Learning further about deep learning. 2.3. Other work not mentioned but should be done. 3. Later Stage <ol style="list-style-type: none"> 3.1. Collecting performance data about training algorithm. 3.2. Employing some visualization method to deliver the result in a more concrete way. 3.3. Preparing for Final Viva by making the system more user-friendly. For example, the system allows user to interact in command line interface or graphics user interface. 3.4. Other issue not mentioned but should be finished. <p>List of background material consulted including World Wide Web pages [1] Airbus Ship Detection Challenge Kaggle. 2018. Airbus Ship Detection Challenge Kaggle. [ONLINE] Available at: https://www.kaggle.com/c/airbus-ship-detection. [Accessed 20 November 2018].</p> |
| 道德规范 Ethics | Please confirm that you have discussed ethical issues with your Supervisor using the ethics checklist on QMPlus. [YES/NO] |

| | |
|---|--|
| | <p>Summary of ethical issues: (put N/A if not applicable)</p> <p>N/A</p> |
| <p>中期目标 Mid-term target.</p> <p>It must be tangible outcomes, E.g. software, hardware or simulation.</p> <p>It will be assessed at the mid-term oral.</p> | <ol style="list-style-type: none"> 1. By the mid-term, the prototype system of one ship detection method should be delivered. The prototype includes: <ol style="list-style-type: none"> 1.1. An executable software prototype; 1.2. A brief visualisation about result of the prototype; 1.3. A list about algorithm performance of the prototype; 2. A keynote that contains all literature information. |

Work Plan (Gantt Chart)

Fill in the sub-tasks and insert a letter X in the cells to show the extent of each task

| | Nov | Dec | Jan | Feb | Mar | Apr | May |
|--|-----|-----|-----|-----|-----|-----|-----|
| Task 1: Early Stage Preparation break down | | | | | | | |
| Early stage research about background on machine learning and deep learning. | X | X | X | | | | |
| Early stage preparation on dataset, including exploration and visualization. | | X | X | X | | | |
| Set up data science environment. | | | X | X | | | |
| Write introduction and part of the method section of the final report. | | | X | X | | | |
| Task 2: Main Stage break down | | | | | | | |
| Work on dataset divide, pre-processing, augmentation. | | | X | X | | | |
| Determine the algorithm, programming language, deep learning frame etc. | | | X | X | X | | |
| Develop prototype of ship detection method and use Agile method to improve it. | | | X | X | X | X | |
| Finish the methods chapter of the report. | | | | | X | X | |
| Task 3: Later Stage break down | | | | | | | |
| Exploring different performance metrics, like the one at Kaggle.com. | | | X | X | | | |
| Evaluate the performance of method 1. | | | | X | X | | |
| Evaluate the performance of method 2. | | | | X | X | | |
| Write results chapter of the report. | | | | | X | | |
| Task 4: Epilogue break down | | | | | | | |
| Compare developed methods. | | | | | X | X | |
| Compare developed methods with methods proposed by others. | | | | | X | X | |
| Make overall analysis on results of developed methods. | | | | | | X | |
| Complete relevant chapter in final report and slides for the final viva. | | | | | | X | |