Configuration Manual

MSc Research Project
MSc in Data Analytics

Sneham Mukherjee
Student ID:x19224826

School of Computing
National College of Ireland

Supervisor: Dr. Barry Haycock

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sneham mukherjee
Student ID:	x19224826
Programme:	MSc in Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Barry Haycock
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	316
Page Count:	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sneham Mukherjee
Date:	31st January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sneham Mukherjee
x19224826

1 Introduction

The aim of this report is to offer a step-by-step instruction to carry out the research work. It contains information regarding the hardware and software configuration, pre-processing steps, model building steps, implementation steps and evaluation.

2 System Configuration

2.1 Hardware Configuration

Device specifications

Yoga Slim 7 14IIL05

Device name	Sneham
Processor	Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz
Installed RAM	16.0 GB (15.8 GB usable)
Device ID	D43D3AFE-CFE2-4C16-838A-61D87238F49A
Product ID	00327-35887-75797-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1: Hardware Configuration

2.2 Software Specification

Software	Specification
Operating System	Windows 10 Home
Programming Language	Python 3.8.5
IDE	Jupyter Notebook
Tools	Ms. Excel, Ms. Word
Report	Overleaf
Web Browser	Google Chrome

Figure 2: Software Specification

3 Data Collection

The dataset is collected from Kaggle[1] which is a openly sourced. It contains images in both .json format and .png format.

Figure 3: 'Ships in Satellite Imagery' dataset

4 Importing Libraries

The following python libraries are imported to carry out the research task.

Importing all libraries to carry out the research code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os, random, cv2, pickle, json, itertools
import imgaug.augmenters as iaa
import imgaug.imgaug
import imutils
from tqdm import tqdm
from imutils.object_detection import non_max_suppression
from IPython.display import SVG
from tensorflow.keras.utils import plot_model, model_to_dot
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from collections import Counter
from sklearn.utils import class_weight
from tqdm import tqdm
from sklearn.preprocessing import LabelBinarizer

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import (Add, Input, Conv2D, Dropout, Activation, BatchNormalization, MaxPooling2D, ZeroPadding2
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint, Callback
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.initializers import *
from tensorflow.keras.models import load_model

from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score

```

Figure 4: List of Libraries

5 Data Loading

Below is the code for loading the image data for preprocessing

```
datasets = [r'C:\Users\Sneham Mukherjee\Thesis\shipsnet']

class_names = ["no-ship","ship"]

class_name_labels = {class_name:i for i,class_name in enumerate(class_names)}

num_classes = len(class_names)
class_name_labels
```

```
{'no-ship': 0, 'ship': 1}
```

```
(images, labels) = load_data()
images.shape, labels.shape
```

```
100%|██████████████████████████████████████████████████| 3000/3000 [00:03<00:00, 935.60it/s]
100%|██████████████████████████████████████████████████| 1000/1000 [00:00<00:00, 1154.98it/s]
```

Figure 5: Loading Data

[1] https://www.kaggle.com/rhammell/ships-in-satellite-imagery

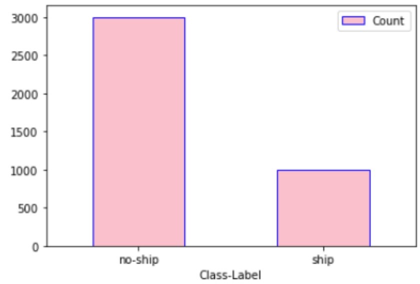
6 EDA of the dataset

The below code is for the EDA of the dataset

```
df.plot.bar(rot=0,color='pink',edgecolor=['blue'])

```

```
<AxesSubplot:xlabel='Class-Label'>
```




```
plt.pie(count,
        explode=(0,0),
        labels=class_names,
        autopct="%1.2f%%",colors=['Violet','Pink'])
plt.axis('equal');
```

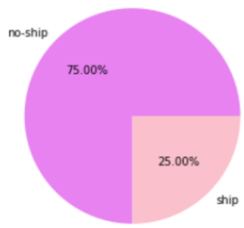


Figure 6: EDA of the dataset

7 Augmentation and Transformation

For Augmentation and Transformation, below code is used.

Augmentation and Transformation Task

```
AUGMENTATION = True
sometimes = lambda aug: iaa.Sometimes(0.5, aug)
seq = iaa.Sequential([
    iaa.Fliplr(0.5),
    sometimes(iaa.OneOf([
                    iaa.EdgeDetect(alpha=(0, 0.7)),
                    iaa.DirectedEdgeDetect(
                        alpha=(0, 0.7), direction=(0.0, 1.0)
                    ),
                ])),
    iaa.Crop(percent=(0,0.1)),
         iaa.LinearContrast((0.75,1.5)),
    iaa.Multiply((0.8,1.2), per_channel=0.2),
    iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.05*255), per_channel=0.5),
    iaa.Affine(
        scale={'x':(0.8,1.2), "y":(0.8,1.2)},
        translate_percent={"x":(-0.2,0.2),"y":(-0.2,0.2)},
        rotate=(-25,25),
        shear=(-8,8)
    )
], random_order=True)


if AUGMENTATION:
    (aug_images, aug_labels) = augment_add(images, seq, labels)
    images = np.concatenate([images, aug_images])
    labels = np.concatenate([labels, aug_labels])
```

```
4000it [00:04, 844.05it/s]
```

Figure 7: Augmentation and Transformation

8 Splitting of Data

The dataset is divided in the ratio mentioned below.

```
total_count = len(images)
total_count

train = int(0.7*total_count)
val = int(0.2*total_count)
test = int(0.1*total_count)

train_images, train_labels = images[:train], labels[:train]
val_images, val_labels = images[train:(val+train)], labels[train:(val+train)]
test_images, test_labels = images[-test:], labels[-test:]

train_images.shape, val_images.shape, test_images.shape
```

Figure 8: Splitting of Data

9 Building Model

Below functions are used to build basic model and conv block.

```
def conv_block(X,k,filters,stage,block,s=2):

    conv_base_name = 'conv_' + str(stage)+block+'_branch'
    bn_base_name = 'bn_'+str(stage)+block+"_branch"

    F1 = filters

    X = Conv2D(filters=F1, kernel_size=(k,k), strides=(s,s),
               padding='same',name=conv_base_name+'2a')(X)
    X = BatchNormalization(name=bn_base_name+'2a')(X)
    X = Activation('relu')(X)

    return X
    pass


def basic_model(input_shape,classes):

    X_input = Input(input_shape)

    X = ZeroPadding2D((5,5))(X_input)

    X = Conv2D(16,(3,3),strides=(2,2),name='conv1',padding="same")(X)
    X = BatchNormalization(name='bn_conv1')(X)

    # stage 2
    X = conv_block(X,3,32,2,block='A',s=1)
    X = MaxPooling2D((2,2))(X)
    X = Dropout(0.25)(X)

#     Stage 3
    X = conv_block(X,5,32,3,block='A',s=2)
    X = MaxPooling2D((2,2))(X)
    X = Dropout(0.25)(X)

#     Stage 4
    X = conv_block(X,3,64,4,block='A',s=1)
    X = MaxPooling2D((2,2))(X)
    X = Dropout(0.25)(X)

#   Output Layer
    X = Flatten()(X)
    X = Dense(64)(X)
    X = Dropout(0.5)(X)

    X = Dense(128)(X)
    X = Activation("relu")(X)

    X = Dense(classes,activation="softmax",name="fc"+str(classes))(X)
```

Figure 9: Building Model

10 Model Implementation

Adam Optimizer is used for the implementation along with loss function binary crossentropy. And the model is trained with 50 epochs with a batch size 16.

```
opt = Adam(lr=1e-3)
model.compile(optimizer=opt,loss='binary_crossentropy',metrics=['accuracy'])
```

Figure 10: Adam Optimizer

```
epochs = 50
batch_size = 16

history = model.fit(train_images,train_labels,
                    steps_per_epoch=len(train_images)//batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(val_images,val_labels),
                    validation_steps=len(val_images)//batch_size,
                    callbacks=[checkpoint, logs]
#
                    )
```

Figure 11: Training Model

11 Model Evaluation

Below code is for the Accuracy/Loss graph for training and validation data

```
def show_final_history(history):

    plt.style.use("ggplot")
    fig, ax = plt.subplots(1,2,figsize=(15,5))
    ax[0].set_title('Loss')
    ax[1].set_title('Accuracy')
    ax[0].plot(history.history['loss'],label='Train Loss')
    ax[0].plot(history.history['val_loss'],label='Validation Loss')
    ax[1].plot(history.history['accuracy'],label='Train Accuracy')
    ax[1].plot(history.history['val_accuracy'],label='Validation Accuracy')

    ax[0].legend(loc='upper right')
    ax[1].legend(loc='lower right')
    plt.show();
    pass
```

Figure 12: Accuracy/Loss graph

Below are the codes for F1 score, Precision, Recall,Accuracy and Confusion Matrix

```
print('The F1 score of Validation Set is',f1_score(val_actual, val_pred, average='weighted'))
print('The Precision score of Validation Set is',precision_score(val_actual, val_pred, average='weighted'))
print('The Recall score of Validation Set is',recall_score(val_actual, val_pred, average='weighted'))
print('The Accuracy Score of Validation Set is',accuracy_score(val_actual, val_pred))

print('The F1 score of Test Set is',f1_score(test_actual, test_pred, average='weighted'))
print('The Precision score of Test Set is',precision_score(test_actual, test_pred, average='weighted'))
print('The Recall score of Test Set is',recall_score(test_actual, test_pred, average='weighted'))
print('The Accuracy Score of Test Set is',accuracy_score(test_actual, test_pred))
```

```
The F1 score of Validation Set is 0.9816674814905351
The Precision score of Validation Set is 0.982017368403507
The Recall score of Validation Set is 0.9816666666666667
The Accuracy Score of Validation Set is 0.9816666666666667
The F1 score of Test Set is 0.9733107241267752
The Precision score of Test Set is 0.9736361502609786
The Recall score of Test Set is 0.9733333333333334
The Accuracy Score of Test Set is 0.9733333333333334
```

Figure 13: F1 score, Precision, Recall and Accuracy

```
val_actual = np.argmax(val_labels,axis=1)

cnf_mat = confusion_matrix(val_actual, val_pred)
np.set_printoptions(precision=2)
sns.heatmap(cnf_mat, annot=True, fmt='d', cmap='BuPu')
plt.title("Confusion Matrix of Validation Set")
plt.figure()
```

Figure 14: Confusion Matrix

Checking Actual vs Predicted on test data

```
rnd_idx = random.sample(range(0,500),10)

class_labels = {i:class_name for (class_name,i) in class_name_labels.items()}
class_labels

for i,idx in enumerate(rnd_idx):

    plt.imshow(test_images[idx])
    plt.title("Actual: {}\nPredicted: {}".format(class_labels[test_actual[idx]],class_labels[test_pred[idx]]))
    plt.grid(None)
    plt.show()
    pass
```

Figure 15: Actual vs Predicted

The code file name is saved as x19224826_Sneham_Mukherjee_Research_Project_Code.ipynb