National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Pavithra Ashokan
Student ID: 22133992

School of Computing
National College of Ireland

Supervisor: Prof. Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Pavithra Ashokan
Student ID:	22133992
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Prof. Hicham Rifai
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	596
Page Count:	8

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:	Pavithra Ashokan
Date:	31st January 2024

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

Pavithra Ashokan
22133992

1 Introduction

Crime detection model is designed to ensure the public safety through the advanced image based classification. Crime is an act that is intended to cause harm either physically or psychologically, that also includes property damage or loss. This leads to punishment by a state or any other authority with respect to the severity of the crime. Shah et al. (2021). This manual is a comprehensive guide to assist users in setting up, configuring and in optimizing the crime detection model for their specific requirements. This configuration manual also aims to empower the users, with all the required information to integrate and utilize the crime detection model.

2 System Requirements

System requirements include hardware and software requirements, that are given in the table below.

2.1 Hardware requirements

OS	Microsoft Windows 11
Processor	12th Gen Intel(R) Core(TM) i5-1235U
RAM	16.0 GB
Storage	256 GB

2.2 Software requirements

Programming Language	Python 3
Tools	Google Colab

3 Dataset Requirements

Dataset consists of train and test datasets, which each 14 categories, Abuse, Arrest, Arson, Assault, Burglary, Explosion, Fighting, Normal videos, Road accidents, Robbery, Shooting, Shoplifting, Stealing and Vandalism.

Displaying: /content/drive/MyDrive/Data Thesis/Test/Arson/Arson041_x264_1120.png

Arson041_x264_1120.png

Displaying: /content/drive/MyDrive/Data Thesis/Test/Arson/Arson007_x264_2950.png

Arson007_x264_2950.png

Figure 1: Arson

Displaying: /content/drive/MyDrive/Data Thesis/Train/Shoplifting/Shoplifting014_x264_51130.png

Shoplifting014_x264_51130.png

Displaying: /content/drive/MyDrive/Data Thesis/Train/Shoplifting/Shoplifting012_x264_15750.png

Shoplifting012_x264_15750.png

Figure 2: Shoplifting

Uploaded the dataset to the Google Colab environment and pre-processed the Crime dataset, by resizing and normalizing the images. Customized and tailored the parameters of the model to the required requirements, with the Google Colab environment.

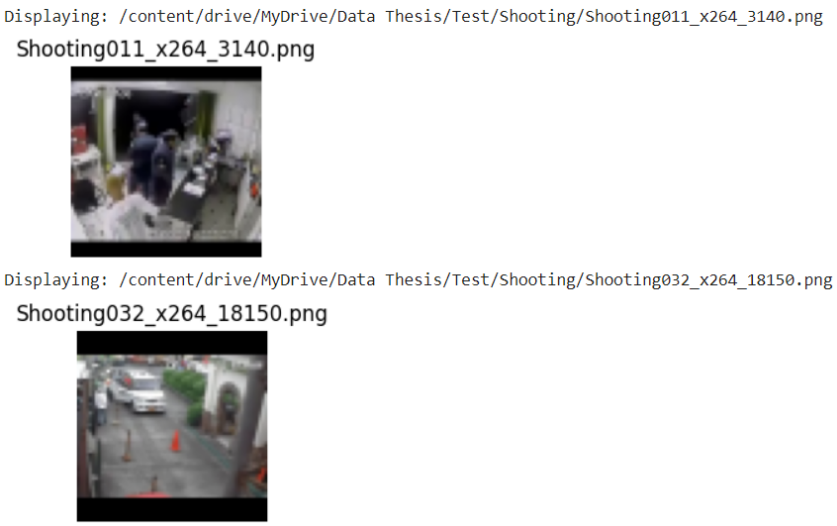

Figure 3: Shooting

4 Model Architecture

The required libraries are installed to evaluate, visualize and to build the layers of the neural networks, such as Pandas, NumPy, Matplotlib, Seaborn, TensorFlow, Keras and more.

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import os

import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory

from tensorflow.keras.applications import DenseNet121
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout,MaxPooling2D , Conv2D,Flatten
from tensorflow.keras.models import Sequential

from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.metrics import classification_report


from IPython.display import clear_output
import warnings
warnings.filterwarnings('ignore')
```

Figure 4: Importing Libraries

Next step involves splitting the training and validation datasets from the training directory and the test dataset are loaded from the separate directory. The image datasets

are then batched to the required batch size and resized to desired dimensions, and the output is obtained containing the file information.

```
Found 345010 files belonging to 14 classes.
Using 276008 files for training.
Found 345010 files belonging to 14 classes.
Using 69002 files for validation.
Found 111308 files belonging to 14 classes.
```

Figure 5: Dataset categories

The Transfer learning model is then developed, using the DenseNet121 architecture as pre-trained model and then the Global Average Pooling 2D layer is added. The final dense layer consists of 'n' units with Softmax activation for the multi-class classification. The model is then compiled using the Adam optimizer.

```
def transfer_learning():
    base_model=DenseNet121(include_top=False,input_shape=INPUT_SHAPE,weights="imagenet")

    thr=149
    for layers in base_model.layers[:thr]:
        layers.trainable=False

    for layers in base_model.layers[thr:]:
        layers.trainable=True

    return base_model
```

```
def create_model():
    model=Sequential()

    base_model=transfer_learning()
    model.add(base_model)

    model.add(GlobalAveragePooling2D())

    model.add(Dense(128, activation="relu"))
    model.add(Dropout(0.2))

    model.add(Dense(n,activation="softmax",name="classification"))

    model.summary()

    return model
```

Figure 6: Transfer Learning Model

5 Model Training

The next step is training the model on the training set and validating it on the validation set. The model training is performed for a specified number of epochs. After this, the true labels ('y_true') and the predicted probabilities ('y_pred') are obtained, for test set. These values are used to evaluate the performance of the model.

```
model=create_model()

model.compile(optimizer="adam",
              loss='categorical_crossentropy',
              metrics = [tf.keras.metrics.AUC()])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 2, 2, 1024)        7037504

 global_average_pooling2d (  (None, 1024)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 128)               131200

 dropout (Dropout)           (None, 128)               0

 classification (Dense)      (None, 14)                1806

=================================================================
Total params: 7170510 (27.35 MB)
Trainable params: 5586254 (21.31 MB)
Non-trainable params: 1584256 (6.04 MB)
```

Figure 7: Model output

```
history = model.fit(x = train_set,validation_data=val_set,epochs = EPOCHS)
```

```
4313/4313 [==============================] - 13915s 3s/step - loss: 0.1914 - auc: 0.9972 - val_loss: 0.0898 - val_auc: 0.9988
```

```
y_true =  np.array([])

for x, y in test_set:
  y_true = np.concatenate([y_true, np.argmax(y.numpy(), axis=-1)])
```

```
y_pred=model.predict(test_set)
```

```
1740/1740 [==============================] - 60s 33ms/step
```

Figure 8: Model Training

```
y_pred
```

```
array([[4.5902032e-02, 1.7751265e-02, 9.4060982e-03, ..., 1.3542435e-03,
        6.9584459e-02, 1.7140590e-02],
       [1.6762480e-02, 8.9247664e-03, 4.2186431e-03, ..., 8.9362811e-04,
        1.4209105e-01, 1.0222074e-02],
       [4.6308540e-02, 5.8252241e-02, 1.5701249e-02, ..., 1.8073161e-03,
        5.1227458e-02, 1.8690057e-02],
       ...,
       [3.0555425e-06, 1.4509319e-08, 4.6502296e-07, ..., 5.0748265e-05,
        1.3541766e-06, 2.1836685e-08],
       [5.4894554e-06, 7.4061792e-09, 4.9845465e-07, ..., 1.8170034e-04,
        1.5264775e-06, 1.3480203e-08],
       [1.5504101e-04, 6.1160634e-08, 4.5755373e-06, ..., 6.1918923e-04,
        3.8409225e-06, 1.9498280e-07]], dtype=float32)
```

```
y_true
```

```
array([ 0.,  0.,  0., ..., 13., 13., 13.])
```

Figure 9: y_pred & y_true

6 Model Evaluation

The model is evaluated with the custom CNN layer on top of the Transfer learning base model. An initial model with limited layers is built and evaluated, as the results are not more efficient. The CNN model is built with additional layers and evaluated with the Adam optimizer. By performing further fine-tuning, the model's best accuracy is obtained.

```
# Create a custom CNN model on top of the transfer learning base model
def create_custom_cnn_model(base_model):
    model = Sequential()
    # Add the base model
    model.add(base_model)
    model.add(GlobalAveragePooling2D())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))

    # Output layer
    model.add(Dense(n, activation='softmax', name='classification'))

    model.summary()
    return model

# Define and compile the base model (transfer learning)
base_model = transfer_learning()

# Update the IMG_HEIGHT and IMG_WIDTH to 64
IMG_HEIGHT = 64
IMG_WIDTH = 64
IMG_SHAPE = (IMG_HEIGHT, IMG_WIDTH, 3)
INPUT_SHAPE = (IMG_HEIGHT, IMG_WIDTH, 3)

# Create a new custom CNN model on top of the transfer learning base model
custom_model = create_custom_cnn_model(base_model)

# Compile the custom model
custom_model.compile(optimizer="adam", loss='categorical_crossentropy', metrics=[tf.keras.metrics.AUC()])

fine_tune_epochs = 3
custom_model.fit(train_set, validation_data=val_set, epochs=fine_tune_epochs)
```

Figure 10: Custom CNN Model

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 2, 2, 1024)        7037504

 global_average_pooling2d_2  (None, 1024)              0
  (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 128)               131200

 dropout_2 (Dropout)         (None, 128)               0

 classification (Dense)      (None, 14)                1806

=================================================================
Total params: 7170510 (27.35 MB)
Trainable params: 5586254 (21.31 MB)
Non-trainable params: 1584256 (6.04 MB)
_________________________________________________________________
Epoch 1/3
4313/4313 [==============================] - 367s 77ms/step - loss: 0.1901 - auc_3: 0.9973 - val_loss: 0.0913 - val_auc_3: 0.9990
Epoch 2/3
4313/4313 [==============================] - 336s 78ms/step - loss: 0.0935 - auc_3: 0.9990 - val_loss: 0.0799 - val_auc_3: 0.9994
Epoch 3/3
4313/4313 [==============================] - 334s 77ms/step - loss: 0.0776 - auc_3: 0.9993 - val_loss: 0.0675 - val_auc_3: 0.9995
<keras.src.callbacks.History at 0x799b52148be0>
```

Figure 11: Model Fine-tuning

The ROC curve is obtained which distinguishes between the different classes and the confusion matrix was also obtained that shows the number of cases for each category of crime for both the actual and predicted.

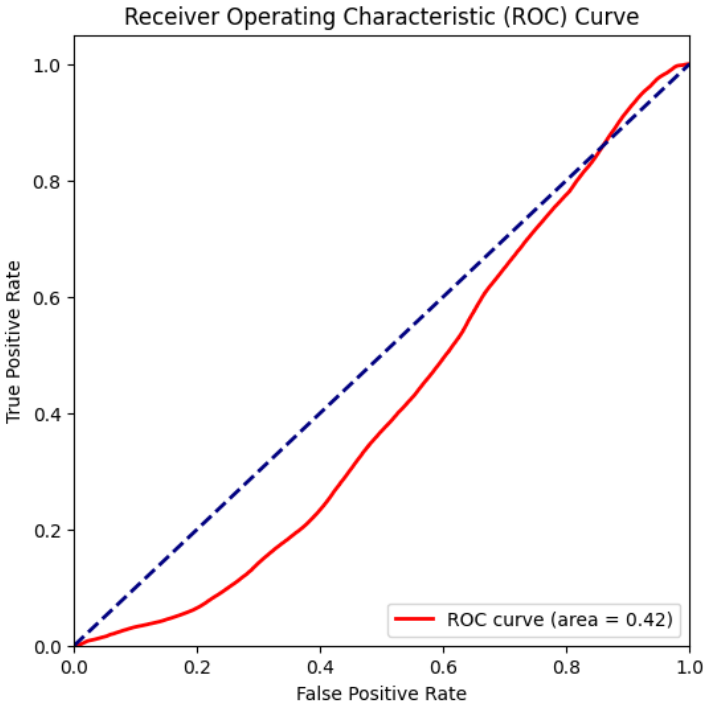

Figure 12: ROC Curve

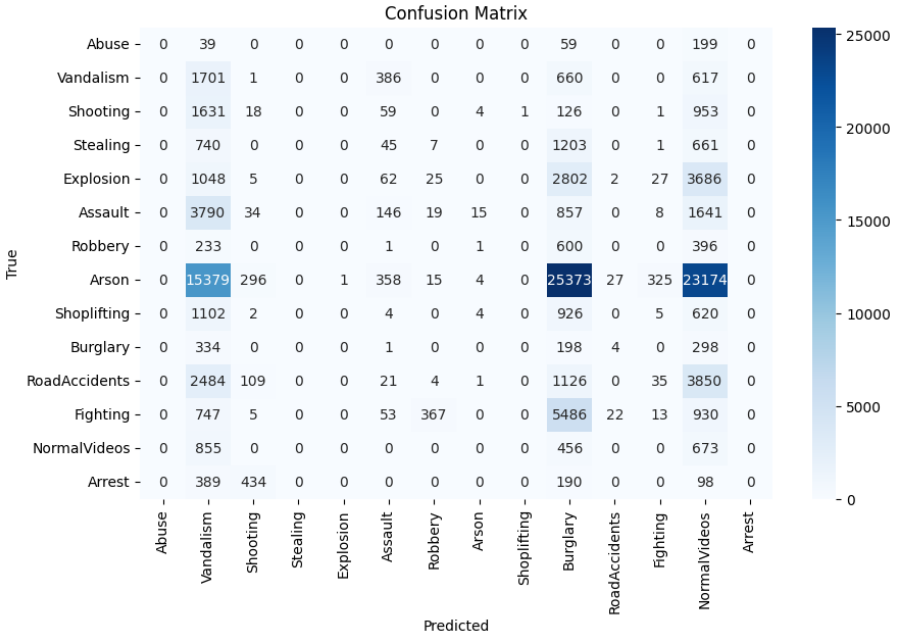

Figure 13: Confusion Matrix

References

Shah, N., Bhagat, N. and Shah, M. (2021). Crime forecasting: A machine learning and computer vision approach to crime prediction and prevention, *Visual Computing for Industry, Biomedicine, and Art* **4**(1): 9.