

# Configuration Manual

MSc Research Project  
Data Analytics

Ibrahim Rinub Babu  
Student ID: X19207387

School of Computing  
National College of Ireland

Supervisor:    Giovani Estrada

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Ibrahim Rinub Babu
<b>Student ID:</b>	X19207387
<b>Programme:</b>	Data Analytics
<b>Year:</b>	2022
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Giovani Estrada
<b>Submission Due Date:</b>	31/01/2022
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	XXX
<b>Page Count:</b>	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.

<b>Signature:</b>	
<b>Date:</b>	31st January 2022

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

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

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

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Ibrahim Rinub Babu  
X19207387

## 1 1. Introduction

The setup manual covers all important information on the research project's software and hardware. It also lists the key libraries that were utilized, and section 3 contains the data description. Furthermore, it explains numerous actions that must be followed in order to recreate the work in any machine that meets the requirements outlined in the following sections. This document explains how to create a Hand gesture Recognition ATM in a clean environment.

## 2 System Configuration

### 2.1 Hardware Requirements

Hardware	Configuration
RAM	24 GB
Hard Disk	1 TB, 256 SSD
CPU	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
GPU	4 GB Nvidia GeForce GTX 1650

Figure 1: Project management objective

### 2.2 Software Requirements

Softwares	Configurations
Operating System	Windows 10
Integrated Development Environment	VS code editor
Cloud environment	Google Colab
language & versions	Python 3.6.9 ,HTML 5, CSS, JS
Frameworks	OpenCV, Flask, Tensorflow, Keras

Figure 2: Project management objective

### 2.2.1 Local Integrated Development Environment

Visual Studio Code by Microsoft is an open-source code editor and compiler for opera Windows, Linux, and macOS. Among the features are debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. The front end of the web application is built with VS code, and the exported model h5 file from the colab is merged with the front end of the website using the Flask framework. Users can customize the theme, keyboard shortcuts, and preferences, as well as install extensions that offer new features.

### 2.2.2 Cloud Integrated Development Environment

Google Colab is a cloud-platform for Collaborator Deep learning research programs. This is akin to a Jupiter notebook environment, which does not require installation and operates totally in the cloud. It makes combining executable code and rich content into a single document easier. It can be used to store images, HTML, LaTeX, and a variety of other files. Figure 1 depicts the appearance of a Google collaboration once you've logged in. 1- Login to Google. 2- Open Google colab and login into it using the email address 3- This online IDE also allows you to import code from GitHub, Google Drive, or just upload it. Any data file you want to use for the learning process and created model can be uploaded to the generated data file using OpenCV. Once the data is mounted on the drive, it can be accessed at any time by simply copying its path.

### 2.2.3 Languages and Framework

To implement the proposed idea, the programming languages such as Python, HTML, CSS, and JS are used. HTML, CSS, and JS are used to implement the front end of the web application, Python Flask is used to host the server and to integrate the model exported as H5 file form google colab.

### 2.2.4 Library's

All the libraries listed below in the figure 3 are imported to execute every task in this research.

Libraries	importing syntax
io	import io
openpyxl	import openpyxl
numpy	import numpy as np
pandas	import pandas as pd
matplotlib	import matplotlib.pyplot as plt
seaborn	import seaborn as sns
tensorflow	import tensorflow as tf from tensorflow.keras.applications.imagenet_utils import preprocess_input, decode_predictions from tensorflow.keras.models import load_model from tensorflow.keras.preprocessing import image from tensorflow import keras
keras	from keras.datasets import cifar10 from keras.preprocessing.image import ImageDataGenerator from keras.callbacks import EarlyStopping, ReduceLROnPlateau from tensorflow.keras.applications.vgg16 import VGG16 from tensorflow.keras.applications.mobilenet import MobileNet from keras.preprocessing.image import ImageDataGenerator from keras.models import Sequential from keras.layers import Dense, Dropout, Activation, Flatten from keras.layers import Conv2D, MaxPool2D
Flask	from flask import Flask, redirect, url_for, request, render_template
werkzeug	from werkzeug.utils import secure_filename
event	from event.pywsgi import WSGIServer

Figure 3: Library Packages

### 3 Dataset Description

OpenCV is used to generate real-time hand gesture and face authentication dataset by accessing the camera for training and testing all the models proposed in this research paper. If the code is executed, it captured 20 images per second. The app.py file should be executed to generate the dataset.

```
import cv2 # importing cv2 library
cam = cv2.VideoCapture(0)
count = 0

check_account_balance = 'check_account_balance'
withdraw_cash = 'withdraw_cash'
next = 'next'
cancel = 'cancel'
Deposit_cash = 'Deposit_cash'

while True:
    ret, img = cam.read()

    cv2.imshow("Test", img)

    if not ret:
        break

    k = cv2.waitKey(1)
    print("Image "+str(count)+"saved")
    file = 'E:\\G_F_images\\Rinub\\'+str(check_account_balance)+str(count)+'.jpg'
    cv2.imwrite(file, img)
    count += 1
    rinub='rinub'
    if count == 120:
        break
cam.release
cv2.destroyAllWindows
```

Figure 4: Open CV

#### 3.1 Hand gesture Datasets

The hand gesture data collection for training the DNN model to perform different jobs in the ATM is created using the OpenCV framework. For hand movements, the webcam's input video stream source is read and processed into individual images. For a single execution of code with the wait key one, it captures 600 photographs of each hand gesture from the background. The display window will open, and the photos will start to be taken. In the directory where the path was assigned, the hand gesture dataset is produced. Back, Cancel, Check Account Balance, Deposit cash, Next, Withdraw Cash are the six directories that were generated based on the labels assigned to them.

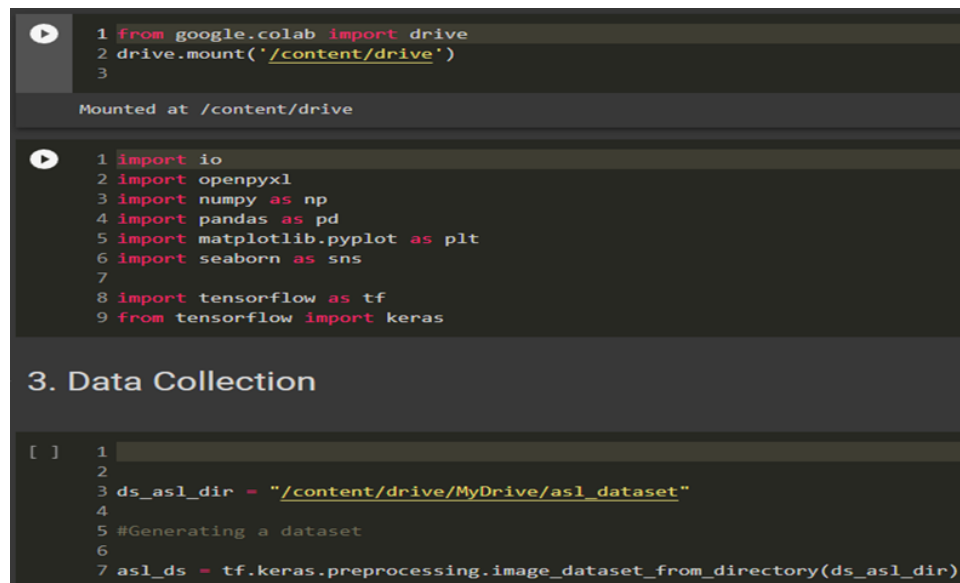
#### 3.2 Face Authentication Datasets

In this data collection, openCV was used to Programmatically construct over 13,000 photographs of faces. On each image, the name of the person pictured has been labelled. There are 400 distinct photos of two of the people in the data set. This data-set will be used to train the model that will be used to authenticate financial transactions 7 at ATMs using facial recognition. The built DNN model assesses numerous elements of your

face, such as eye placement and nose width, and combines all of this data into a single code that identifies and authenticates you.

## 4 Environment Setup

**Step1** A google colab notebook is opened in the name Hand\_Gesture\_Recognition\_ATM.ipynb. The generated data-set is uploaded to the google drive, this is done to give google colab access to the code. In the figure 1, shows that the data in the google drive is accesses programmatically in the google colab using drive library.



```
1 from google.colab import drive
2 drive.mount('/content/drive')
3
Mounted at /content/drive

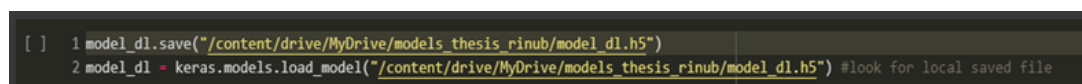
1 import io
2 import openpyxl
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7
8 import tensorflow as tf
9 from tensorflow import keras

3. Data Collection

[ ] 1
     2
     3 ds_asl_dir = "/content/drive/MyDrive/asl_dataset"
     4
     5 #Generating a dataset
     6
     7 asl_ds = tf.keras.preprocessing.image_dataset_from_directory(ds_asl_dir)
```

Figure 5: Data collection

**Step 2** After building different DNN models such as custom CNN and 3 transfer learning models such as MobileNet, VGG16 and ResNet50. The best model (Resnet50) is selected with different evaluation matrices and factors. And exported as H5 file which is model\_dl.h5. The exported model\_dl.h5 file is generated in google drive in the selected path as shown in the figure 2.



```
[ ] 1 model_dl.save("/content/drive/MyDrive/models_thesis_rinub/model_dl.h5")
     2 model_dl = keras.models.load_model("/content/drive/MyDrive/models_thesis_rinub/model_dl.h5") #look for local saved file
```

Figure 6: Model path

**Step 3** The exported model\_dl.h5 file is integrated with the front-end web application with the flask framework. This is done by loading the h5 file in app.py file by giving its directory. The code snippet is shown in the below figure 3.

```

# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

# Define a flask app
app = Flask(__name__)

# # Model saved with Keras model.save()
MODEL_PATH = 'model_dl.h5'

# # Load your trained model
model = load_model(MODEL_PATH)
# model._make_predict_function() # Necessary

def model_predict(img_path, model):
    img = image.load_img(img_path, target_size=(224, 224))

    # Preprocessing the image
    x = image.img_to_array(img)
    # x = np.true_divide(x, 255)
    x = np.expand_dims(x, axis=0)
    img = np.vstack([x])
    classes = model.predict(img, batch_size=256)
    probabilities = model.predict_proba(img, batch_size= 256)
    probabilities_formatted = list(map("{:.2f}%".format, probabilities[0]*100))
    print(probabilities_formatted)
    return probabilities_formatted

```

Figure 7: Flask Integration

**Step 4** The file structure of the micro web application framework is shown in the below figure 9. The h.5 file is loaded with this file structure in the VS code editor and the file path is rendered in the app.py file. while executing the app.py file the web application gets hosted in the local browser which is shown in the figure8.

```

* Restarting with windowsapi reloader
2021-12-16 06:45:22.468780: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
to enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Debugger is active!
* Debugger PIN: 141-027-580
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
WARNING:tensorflow:From c:\Users\Asus TUF\Downloads\DL-Project-For-Beginner-master\app.py:40: Sequential.predict_proba (from tensorflow.python.keras
engine.sequential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:

```

Figure 8: Hosting Local Server

## 5 Training and Evaluating the model

**Step 1** After accessing the data, some of the important aspects are carried out such as data preprocessing and data augmentation. Using the library imagedatagenerator from keras all the above-mentioned process are carried which is clearly shown in the below figure 2.

**Step 2** After data preprocessing and splitting the image data set for testing and training. The model has been trained and evaluated using different matrices which is shown in the

```

1 #Augmenting the images
2
3 from keras.preprocessing.image import ImageDataGenerator
4 data_augmentation = ImageDataGenerator(rotation_range=15, rescale=1/255, zoom_range=0.1, horizontal_flip=True,
5                                       width_shift_range=0.1, height_shift_range=0.1, validation_split=0.2)
6
7 #Setting train/test split
8
9 asl_train_ds = data_augmentation.flow_from_directory(directory="/content/drive/MyDrive/asl_dataset", target_size=(64, 64),
10                                                    class_mode="categorical", batch_size=64, subset="training")
11 asl_test_ds = data_augmentation.flow_from_directory(directory="/content/drive/MyDrive/asl_dataset", target_size=(64, 64),
12                                                    class_mode="categorical", batch_size=64, subset="validation")

```

Found 1152 images belonging to 5 classes.  
Found 288 images belonging to 5 classes.

Figure 9: Data Pre-processing

figure 10. All the 4 models matrices are taken into consideration. By comparing the epochs, Precision, recall, validation accuracy and number of epochs, run time and real time performance using OpenCV the best model is exported. After evaluating, the best model is exported as model\_dl.h5 file.

```

1 #OPT = tensorflow.keras.optimizers.Adam(lr=0.0001)
2 base_learning_rate = 0.0001
3
4 vgg_model.compile(loss='binary_crossentropy',
5                  metrics=['accuracy', 'Precision', 'Recall'],
6                  optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate))

```

```

[ ] 1 model_history1=vgg_model.fit(train_dataset,
2                               validation_data=validation_dataset,
3                               epochs = 5)

```

Figure 10: Model Export

## 6 Making Predictions

The exported model\_dl.h5 file is integrated with the OpenCV framework. The OpenCV.py file should be executed to capture live image and loaded to the exported ResNet50 model. The predicted value is shown as the message in the front end.

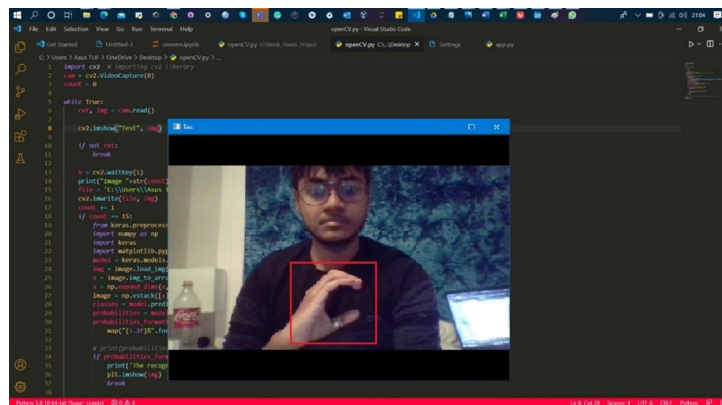


Figure 11: Prediction