Configuration Manual

MSc Research Project
Data Analytics

AKSHAY MENON
Student ID: x21173036

School of Computing
National College of Ireland

Supervisor: Dr.Paul Stynes Dr.Pramod Pathak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	AKSHAY MENON
Student ID:	x21173036
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr.Paul Stynes Dr.Pramod Pathak
Submission Due Date:	01/02/2023
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	10

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:	
Date:	1st February 2023

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

AKSHAY MENON
x21173036

1 Initial Environment Setting

The software specification for the setup are described below in the Figure 1. :-

Programming language	Python (v3.9)
Cloud Platform	Google drive for storage
Device configuration	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
GPU UTILIZATION	K 80 with 2xvCPU of 12 GB
CPU UTILIZATION	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz With 8GB
IDE	Jupyter Notebook,Google Colab
Image Labelling Tool	Microsoft Visual Object Tagging Tool

Figure 1: Environment Configuration

Once the envinronment with all valid specification has been setup we will be needing Jupyter notebook and Google Colab to run first module of the reseach project which is shuttlecock tracking .

1.1 Initiating Environment

Let us first initiate by running module on jupyter notebook.
1.Install and import all libraries for running shuttlecock tracking module which requires frame to image conversion as pre-requisite in Fig 2 .

```
!pip install argparse
!pip install numpy
!pip install matplotlib
!pip install pillow
!pip install h5py
!pip install pydot
!pip install keras
!pip install tensorflow-gpu
!pip install opencv-python
```

Figure 2: Model Libraries

2. The Frames from the video are converted into png format through the script in Fig 3.

```python
import cv2
import csv
import os
import sys
import shutil
try:
    videoName = sys.argv[1]###########input path###########
    outputPath = sys.argv[2]###########output Path###########
    if (not videoName) or (not outputPath):
        raise ''
except:
    print('usage: python3 Frame_Generator.py <videoPath> <outputFolder>')
    exit(1)
if outputPath[-1] != '/':
    outputPath += '/'
    
if os.path.exists(outputPath):
    shutil.rmtree(outputPath)
os.makedirs(outputPath)
#Segment the video into frames
cap = cv2.VideoCapture(videoName)
success, count = True, 0
success, image = cap.read()
while success:
    cv2.imwrite(outputPath + '%d.png' %(count), image)
    count += 1
    success, image = cap.read()
```

Figure 3: Frame generation

3. The generated frames are stored in local drive which is used for image labelling using Microsoft visual object tagging tool and saving output in csv format which has coordinates of the shuttlecock in the frame in Fig 4.

Figure 4: Image labelling

4. The labelled images are used for annotation using Heat Map prediction in Fig 5.

Figure 5: Export to CSV

```
#1.Create heatmap as Ground Truth
import glob
import csv
import numpy
import matplotlib.pyplot as plt
from PIL import Image
import os

size = 20
#create gussian heatmap
def gaussian_kernel(variance):
    x, y = numpy.mgrid[-size:size+1, -size:size+1]
    g = numpy.exp(-(x**2+y**2)/float(2*variance))
    return g


#make the Gaussian by calling the function
variance = 10
gaussian_kernel_array = gaussian_kernel(variance)
#rescale the value to 0-255
print(int((len(gaussian_kernel_array)+1)/2))
gaussian_kernel_array =  gaussian_kernel_array * 255/gaussian_kernel_array[int((len(gaussian_kernel_array)+1)/2)][int((len(gaussi
#change type as integer
gaussian_kernel_array = gaussian_kernel_array.astype(int)

#show heatmap
plt.imshow(gaussian_kernel_array, cmap=plt.get_cmap('gray'), interpolation='nearest')
plt.colorbar()
```

Figure 6: Heat Map Prediction

5. The Labelled Csv file and Heat map prediction are used to generate training and testing numpy array file in Fig 6 .

```python
from glob import glob
import piexif
from keras_preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
import pandas as pd
from sklearn.model_selection import train_test_split
HEIGHT=288
WIDTH=512
mag = 1
sigma = 2.5
def genHeatMap(w, h, cx, cy, r, mag):
    if cx < 0 or cy < 0:
        return np.zeros((h, w))
    x, y = np.meshgrid(np.linspace(1, w, w), np.linspace(1, h, h))
    heatmap = ((y - (cy + 1))**2) + ((x - (cx + 1))**2)
    heatmap[heatmap <= r**2] = 1
    heatmap[heatmap > r**2] = 0
    return heatmap*mag
try:
    (opts, args) = getopt.getopt(sys.argv[1:], '', [
        'batch=',
        'label=',
        'frameDir=',
        'dataDir='
    ])
    if len(opts) != 4:
        raise ''
except:
    print('usage: python3 gen_data.py --batch=<batchSize> --label=<csvFile> --frameDir=<frameDirectory> --dataDir=<npyDataDirecto
    exit(1)
batch = 500
labelPath = 'C:/Users/JAYA MENON/OneDrive - Livent Corporation/Desktop/DAPA-CA2/rally_video/video/frame/frames-20221127T112814Z-0
frameDir = 'C:/Users/JAYA MENON/OneDrive - Livent Corporation/Desktop/DAPA-CA2/rally_video/video/frame/frames-20221127T112814Z-00
dataDir = 'C:/Users/JAYA MENON/OneDrive - Livent Corporation/Desktop/DAPA-CA2/rally_video/video/frame/frames-20221127T112814Z-001
for (opt, arg) in opts:
    if opt == '--batch':
        batch = int(arg)
    elif opt == '--label':
        labelPath = arg
    elif opt == '--frameDir':
        frameDir = arg
    elif opt == '--dataDir':
        dataDir = arg
    else:
        print('usage: python3 gen_data.py --batch=<batchSize> --label=<csvFile> --frameDir=<frameDirectory> --dataDir=<npyDataDir
        exit(1)
p = os.path.abspath(os.path.join(frameDir, '1.png'))
img = img_to_array(load_img(p))
```

Figure 7: Numpy Array files generation

6.The training of the shuttlecock will be done on google colab with K8 GPU memory for the same once initiating Google colab notebook change runtime to GPU Kosaian et al. (2021) and mount drive on your google drive in Fig 7 and Fig 8.

7. Upload NPY from local device to google drive for training purpose in Fig 9.

8. From the Google colab run the TrackNet model script in Fig 10.

9.The Model training script needs to be run using labelled csv files and Numpy array files at epoch 30 with tolerance 4 with folder for saving weight in Fig 11.

10. The trained images are predicted using saving weight for the accuracy script has to be run in Fig 12.

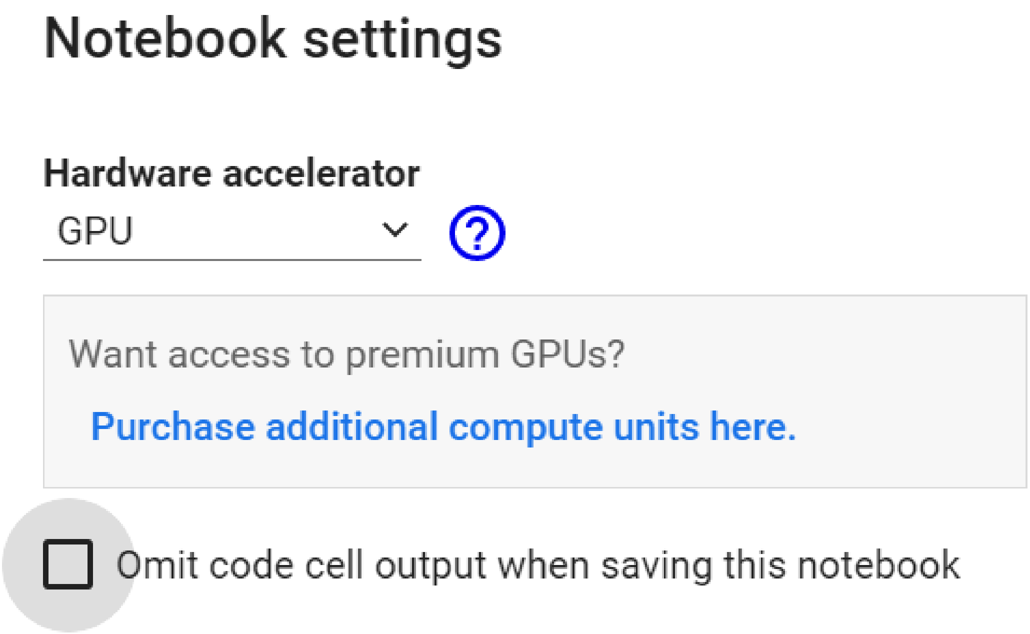

Figure 8: GPU

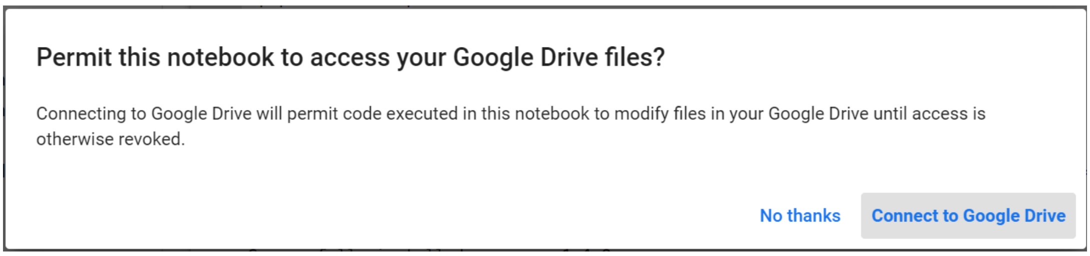

Figure 9: Mount Google Drive

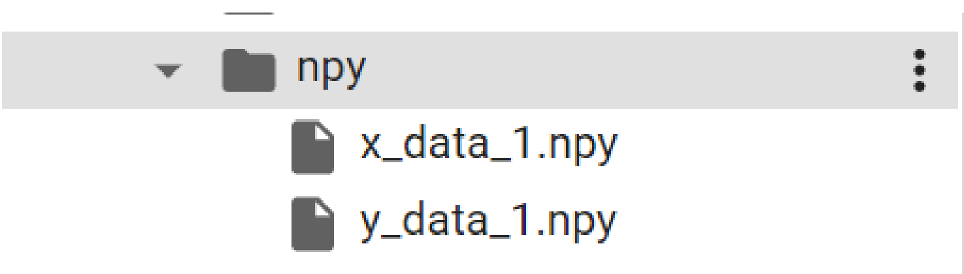

Figure 10: Npy file uploading to G drive

TrackNet Model loading

```
[ ]  from keras.models import *
     from keras.layers import *
     from keras.activations import *

     def TrackNet( input_height, input_width ): #input_height = 288, input_width = 512

       imgs_input = Input(shape=(9,input_height,input_width))
```

Figure 11: TrackNet Model loading

```
try:
  (opts, args) = getopt.getopt(sys.argv[1:], '', [
    'load_weights=',
    'save_weights=',
    'dataDir=',
    'epochs=',
    'tol='
  ])
  if len(opts) < 4:
    raise ''
except:
  print('usage: python3 train_TrackNet.py --load_weights=<previousWeightPath> --save_weights=<newWeightPath> -
  print('argument --load_weights is required only if you want to retrain the model')
  exit(1)

paramCount={
  'load_weights': 0
```

Figure 12: Training Images

Model Accuracy Prediction

```
[ ]  !python3 '/content/TrackNetv2/3_in_1_out/accuracy.py' --load_weights='/content/TrackNetv2/3_in_1_out/model_33'

     2022-12-14 23:36:44.692323: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is opt
     To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
     2022-12-14 23:36:45.620814: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Coulc
     2022-12-14 23:36:45.620925: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Coulc
     2022-12-14 23:36:45.620944: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot dlc
     2022-12-14 23:36:48.733506: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:42] Overriding orig_valu
     Beginning evaluating......
     ==========================================================
     2022-12-14 23:36:51.323216: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 828112896 excee
     156/156 [==============================] - 13s 51ms/step
     Finish evaluating data1:(TP, TN, FP1, FP2, FN)=(134, 7, 1, 3, 11)
     ==========================================================
     Number of true positive: 134
     Number of true negative: 7
     Number of false positive FP1: 1
     Number of false positive FP2: 3
     Number of false negative: 11
     accuracy: 0.9038461538461539
```

Figure 13: Model Accuracy Prediction

2 Initiating Environment for Player Service Fault Detection

1.The Machine learning framework for player pose detection need to install following libraries including google mediapipe pose Lugaresi et al. (2019) in fig 14.

```
# Player Service Fault Detection
```

```
import cv2
from cv2 import destroyAllWindows
import mediapipe as mp
import numpy as np
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_pose = mp.solutions.pose
```

Figure 14: Media pipe Library

2. Post implementing relevant libraries Youtube video is uploaded into the script using OpenCV to read frame and generate body landmark in fig 15.

```
cap = cv2.VideoCapture('C:/Users/JAYA MENON/OneDrive - Livent Corporation/Desktop/DAPA-CA2/rally_video/video/frame/Recording #4.m
 # Check if video path is correct else apply correct path
if (cap.isOpened()== False):
  print("Error opening video stream or file")
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while (cap.isOpened()):
        ret, frame = cap.read()
        if ret == True:

        # Recolor image to RGB
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make detection
        results = pose.process(image)

        # Recolor back to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Extract landmarks
        try:
            landmarks = results.pose_landmarks.landmark
            print(landmarks)

            # Extract Pose Landmarks
            pose = results.pose_landmarks.landmark
            pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten()
```

Figure 15: mediapipe script

3. We have made logical expression for calculating angle from mediapipe script in fig 16.

```
# Visualize angle
cv2.putText(image, str(knee_l),
               tuple(np.multiply(KNEE, [640, 480]).astype(int)),
               cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA
                    )

# Curl counter logic
if angle_left > 150:
    stage = "Foot Not stationary"
else :
    stage="Not Foul"
    #print (stage)
```

Figure 16: calculate angle

```
0.6717962622642517,
0.3016737997531891,
0.06349479407072067,
0.9991064667701721,
0.6839492321014404,
0.42448103427886963,
-0.27126258611679077,
0.9885246157646179,
0.6692742109298706,
0.41925638914108276,
0.09628592431545258,
0.09622182697057724,
0.6670956611633301,
0.5342804193496704,
-0.2637346684932709,
0.9472872614860535,
0.6531014442443848,
0.5067652463912964,
0.03243167698383331,
```

```
In [262]: row
```

```
0.01844826340675354,
0.22507211565971375,
0.6746513247489929,
0.513401985168457,
-0.09573377668857574,
0.9992747902870178,
0.6721502542495728,
0.5115118622779846,
0.09570913016796112,
0.997275710105896,
0.6689900755882263,
0.6765620112419128,
-0.10818184912204742,
0.8476671576499939,
0.6872453093528748,
0.6702944040298462,
0.1266680657863617,
0.416667103767395,
0.6694719791412354,
```

Figure 17: Body Landmark Generation

4. Body Landmark is generated and recorded in fig 17.
5. The generated landmark in the form of data frame undergoes training in fig 18.

```
73]: from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

74]: from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

Figure 18: Training of Body Landmark

6. Model training and evaluation to check best fit in fig 19.

7.The best model has been selected as Random Classifier which will used for saving model and for predicting outcome of the machine learning framework in fig 20.

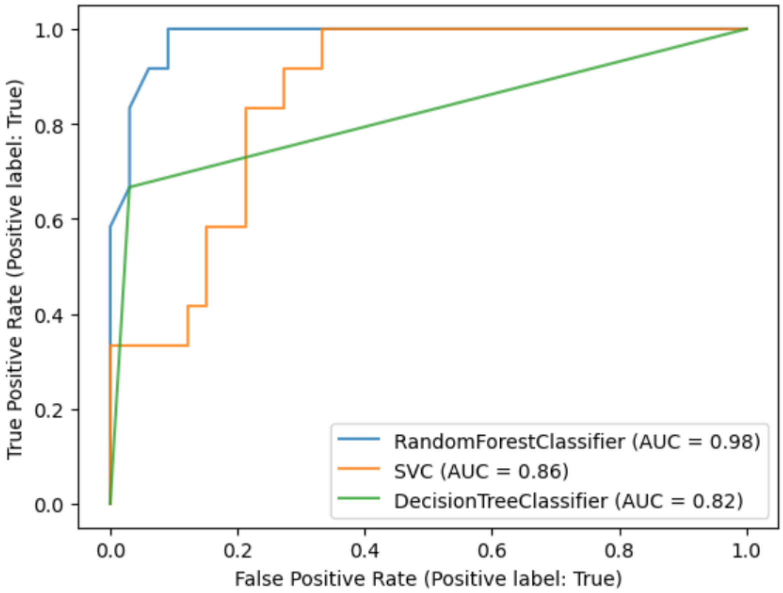

Figure 19: Best Model Fit

```
# Concate rows
row = pose_row+hand_row

  # Append class name
  row.insert(0, class_name)

  # Export to CSV
  with open('coords.csv', mode='a', newline='') as f:
      csv_writer = csv.writer(f, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
      csv_writer.writerow(row)

# Make Detections
X = pd.DataFrame([row])
service_fault_class = model.predict(X)[0]
service_fault_prob = model.predict_proba(X)[0]
print(service_fault_class, service_fault_prob)

# Grab ear coords
coords = tuple(np.multiply(
                np.array(
                    (results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_KNEE].x,
                     results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_KNEE].y))
            , [640,480]).astype(int))

cv2.rectangle(image,
              (coords[0], coords[1]+5),
              (coords[0]+len(body_language_class)*20, coords[1]-30),
              (245, 117, 16), -1)
cv2.putText(image, body_language_class, coords,
            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

# Get status box
cv2.rectangle(image, (0,0), (250, 60), (245, 117, 16), -1)

# Display Class
cv2.putText(image, 'CLASS'
            , (95,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, body_language_class.split(' ')[0]
            , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
```

Figure 20: Test using Saved model

References

Kosaian, J., Phanishayee, A., Philipose, M., Dey, D. and Vinayak, R. (2021). Boosting the throughput and accelerator utilization of specialized cnn inference beyond increasing batch size, *International Conference on Machine Learning*, PMLR, pp. 5731–5741.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J. et al. (2019). Mediapipe: A framework for building perception pipelines, *arXiv preprint arXiv:1906.08172* .