PROJECT REPORT

INTRODUCTION :

Pose Detection is a subset of the Computer Vision (CV) technique that predicts the tracks and location of a person or object. This is done by looking at the combination of the poses and the direction of the given person or object.

1.2 PURPOSE :

Earlier we got a better understanding of Pose Detection where we built a model using a pre-trained model. There are many interesting applications and use cases of pose detection. Now, in this article, we'll discuss one such interesting application and build a model to solve that problem.

The objective of this article is to build a model that can classify the cricket shots using the pose of a player. For this, an image will be input into the model. It will detect the pose of the person in the image and then using the pose that was detected, we will classify what type of shot it was.

2. LITERATURE SURVEY :

2.1 EXISTING PROBLEM :

PROJECT REPORT

For various cricketing academies and cricket conducting boards , it would be very difficult to estimate the shot that was played by the players in the cricket matches. To overcome this regard of problems , we / respective needed can use this cricket pose estimation software which can tell the accurate value of the shot played by the player by using its past experiences and learning data sets.

2.2 PROPOSED SOLUTION :

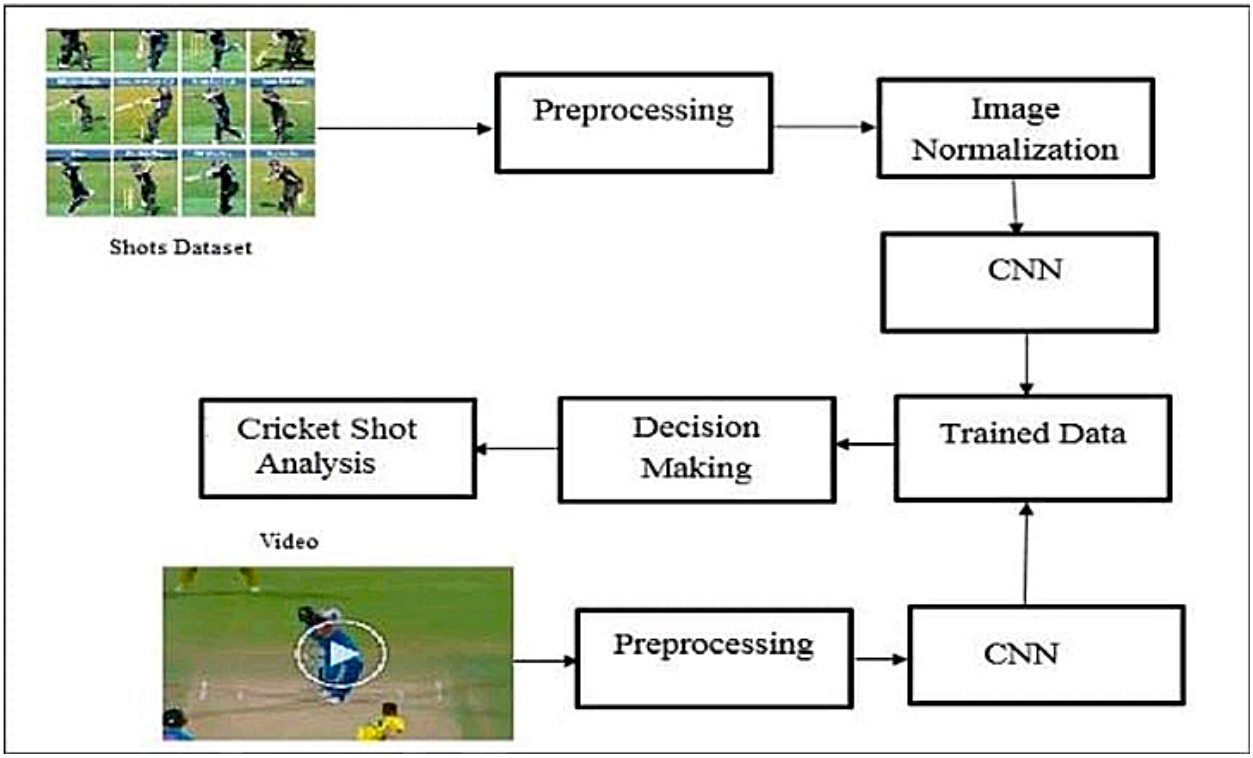
We can use this pose estimation software to overcome the above mentioned problem.This methodology works depending upon the learning datasets along with the past experiences.

This deep learning project model would give the name of cricket shot played by the player in the match with an average accuracy between 70-80.

3. THEORITICAL ANALYSIS:

PROJECT REPORT

3.1 BLOCK DIAGRAM :



3.2 HARDWARE/SOFTWARE DESIGNING :

1. Install Dependencies

2. Load and pre-process the data

3. Data Augmentation

4. Detecting pose using detectron2

5. Classifying cricket shot using pose of a player

6. Evaluating model performance

4. EXPERIMENTAL INVESTIGATIONS :

First, we download tf_pose_estimate from the GitHub repository. Next, we import some necessary libraries:

```
import argparse
import logging
import sys
import time

from tf_pose import common
from tqdm import tqdm
import cv2
import os
import pandas as pd
import numpy as np
from tf_pose.estimator import TfPoseEstimator
from tf_pose.networks import get_graph_path, model_wh
import matplotlib.pyplot as plt
%matplotlib inline
```

We then get the '*mobilenet_thin'* model:

```
size = '432x368'
model = 'mobilenet_thin'


w, h = model_wh(size)
if w == 0 or h == 0:
    e = TfPoseEstimator(get_graph_path(model), target_size=(432, 368))
else:
    e = TfPoseEstimator(get_graph_path(model), target_size=(w, h))
```

Once this is done, we explore the data from the 3 folders for *CUT, SWEEP and DRIVE* respectively. I have displayed the code for *CUT,* the procedure remains the same for *SWEEP* and *DRIVE.*

Next, we extract the *x* and *y* coordinates of the 18 points identified and append it to a list for each category. We then convert it into a Pandas Data-frame and display the first 5 values to get an insight on the data.

As we see on scrolling down in the above cell, the output is the *x* and *y* coordinates of the 18 points (0 – 17). We perform the exact same operation for the other 2 categories.

We then add another label to the dataframe (named *pose* here) and set it to 0

for Sweep shots, 1 for Cut shots and 2 for Drive shots to help us classify later. We then normalize all the *x* and *y* values from the dataframe. Any suitable technique can be used, one such is min-max normalization which I shall briefly explain here.

Min-Max Normalization

Here, for every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1.

It can be calculated using this formulae for every feature. This ensures that the values are between 0 and 1.

For a practical case, it can be visualized as shown below. I've taken the example from here. You can read more about the same there. There are multiple methods of normalization, this is one of the simplest and most used methods.

After normalization, we fill in the *NaN* values and finally build the

classification model.

5. FLOWCHART :

We have got a flowchart which we made analysis throughout the project.

It helped us very much throughout the project making.

INSTALL DEPENDENCIES

LOAD AND PRE-PROCESS THE DATA

DATA AUGMENTATION

DETECTING POSE USING DETECTRON2

CLASSIFYING CRICKET SHOT USING POSE OF A PLAYER

EVALUATING MODEL PERFORMANCE

6. RESULT :

PROJECT REPORT

PROJECT REPORT

PROJECT REPORT

PROJECT REPORT

7. ADVANTAGES AND DISADVANTAGES :

ADVANTAGES :

1. Many cricketing boards and academies can easily determine the cricket shot that was played by the players.
2. Much time will be saved .
3. More cost efficiency.
4. Coaches and team heads can make a analysis of a particular player by keeping a record of shots played by him,

DISADVANTAGES :

1. Other teams also can make analysis of the opponent players and can find the player’s weaknesses.
2. This project can also be used in a negative way.

3. 100% accuracy cannot be maintained all times.

8. APPLICATIONS :

1. This can be used in various cricketing academies / boards.

2. Other sports boards can also use this software by loading their own Learning datasets.

3. This can also be used for various types of analysis.

9. CONCLUSION :

Here I use a random forest classifier on top of this 18 dimensional feature space to classify each instance as one of the three shots. I use a 70/30 split of train and test data.

In order to improve the accuracy, you can play around with different hyperparameters like increasing the number of hidden layers in the model, changing the optimizer, changing the activation function, increasing the

number of epochs, and much more.

I hope you are already familiar with the hyperparameter tuning for the neural networks do try them out at your end and share your performance in the comment section. So this is how we can build a model to classify the shots using the pose of a player.

10. FUTURE SCOPE :

This project is just in the starting stage . Future Enhancements can be made in the project in many ways. Many more datasets can be uploaded inorder to improve the accuracy.

This project may be upgraded based on the need of the project in the future.

11. BIBILOGRAPHY :

PROJECT REPORT

SOURCE CODE :

```
# -*- coding: utf-8 -*-

"""Running_Pose_Estimate_1.ipynb



Automatically generated by Colaboratory.



Original file is located at

    https://colab.research.google.com/drive/1nLOw9phwTviv1CBK4z4GSPMCIVROphrM

"""



# Jovian Commit Essentials

# Please retain and execute this cell without modifying the contents
```

PROJECT REPORT

for `jovian.commit` to work

!pip install jovian --upgrade -q

import jovian

jovian.set_project('cricketshot')

jovian.set_colab_id('1nLOw9phwTviv1CBK4z4GSPMCIVROphrM')

!pip install -qq git+https://www.github.com/ildoonet/tf-pose-estimation

!pip install -qq pycocotools

!pip install jovian

 # Commented out IPython magic to ensure Python compatibility.

 # %load_ext autoreload

PROJECT REPORT

```
# %autoreload 2

import seaborn as sns

import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = (8, 8)

plt.rcParams["figure.dpi"] = 125

plt.rcParams["font.size"] = 14

plt.rcParams['font.family'] = ['sans-serif']

plt.rcParams['font.sans-serif'] = ['DejaVu Sans']

plt.style.use('ggplot')

sns.set_style("whitegrid", {'axes.grid': False})



# Commented out IPython magic to ensure Python compatibility.

import argparse

import logging
```

PROJECT REPORT

```
import sys

import time



from tf_pose import common

from tqdm import tqdm

import cv2

import os

import pandas as pd

import numpy as np

from tf_pose.estimator import TfPoseEstimator

from tf_pose.networks import get_graph_path, model_wh

import matplotlib.pyplot as plt

# %matplotlib inline
```

PROJECT REPORT

```
size = '432x368'

model = 'mobilenet_thin'



w, h = model_wh(size)

if w == 0 or h == 0:

    e = TfPoseEstimator(get_graph_path(model), target_size=(432,

368))

else:

    e = TfPoseEstimator(get_graph_path(model), target_size=(w, h))



cutDir = '../input/cricket/cut/cut/'

cutimages = os.listdir(cutDir)

print(len(cutimages))
```

PROJECT REPORT

```
cutShots = []

cutFiles=[]

pbar = tqdm(total=len(cutimages))

for img in cutimages:

    image = common.read_imgfile(cutDir+img, None, None)

    humans = e.inference(npimg = image, upsample_size=4.0)

    if len(humans)>0:

        cutShots.append(humans[0])

        cutFiles.append(image)

    pbar.update(1)

pbar.close()

print("Cut Shot Examples: ",len(cutFiles))



sweepDir = '../input/cricket/sweep/sweep/'
```

```
sweepimages = os.listdir(sweepDir)

print(len(sweepimages))



sweepShots = []

sweepFiles = []

pbar = tqdm(total=len(sweepimages))

for img in sweepimages:

    image = common.read_imgfile(sweepDir+img, None, None)

    humans = e.inference(image, resize_to_default=(w > 0 and h > 0),
upsample_size=4.0)

    if len(humans)>0:

        sweepShots.append(humans[0])

        sweepFiles.append(image)

        pbar.update(1)
```

```
pbar.close()

print("Sweep Examples: ",len(sweepFiles))



driveDir = '../input/cricket/drive/drive/'

driveimages = os.listdir(driveDir)

print(len(driveimages))



driveShots = []

driveFiles = []

pbar = tqdm(total=len(driveimages))

for img in driveimages:

    image = common.read_imgfile(driveDir+img, None, None)

    humans = e.inference(image, resize_to_default=(w > 0 and h > 0),
```

```
upsample_size=4.0)

    if len(humans)>0:

        driveShots.append(humans[0])

        driveFiles.append(image)

    pbar.update(1)



pbar.close()

print("Drive Examples: ",len(driveFiles))



def humanToDict(hum):

    resultDict = {}

    parts = hum.body_parts.keys()

    for p in parts:

        resultDict[str(p)+'_x'] = hum.body_parts[p].x
```

```
        resultDict[str(p)+'_y'] = hum.body_parts[p].y

        #resultDict[str(p)+'_p'] = hum.body_parts[p].p

    return resultDict



cutList = []

for sh in cutShots:

    cutList.append(humanToDict(sh))



cutdf = pd.DataFrame(cutList)

#standHumadf['img'] = standFiles

cutdf.head()



sweepList = []

for jh in sweepShots:
```

```
    sweepList.append(humanToDict(jh))



sweepdf = pd.DataFrame(sweepList)

sweepdf.head()



driveList = []

for jh in driveShots:

    driveList.append(humanToDict(jh))



drivedf = pd.DataFrame(driveList)

drivedf.head()



sweepdf.shape, cutdf.shape, drivedf.shape,
```

PROJECT REPORT

sweepdf['pose']=0

cutdf['pose']=1

drivedf['pose']=2

alldata = sweepdf.append(cutdf)

alldata = alldata.append(drivedf)

alldata = alldata.reset_index(drop = True)

alldata.shape

allFiles = sweepFiles + cutFiles + driveFiles

allShots = sweepShots + cutShots + driveShots

def min_max_normalize(df):

PROJECT REPORT

```
    xcols = [c for c in df.columns if 'x' in c]

    xdf = df[xcols]

    xdf = xdf.subtract(xdf.min(axis=1), axis=0)

    xdf = xdf.divide(xdf.max(axis=1), axis=0)



    ycols = [c for c in df.columns if 'y' in c]

    ydf = df[ycols]

    ydf = ydf.subtract(ydf.min(axis=1), axis=0)

    ydf = ydf.divide(ydf.max(axis=1), axis=0)

    if 'pose' in df.columns:

        resultdf =  pd.concat( [xdf,ydf, df[['pose']]], axis=1 )

    else:

        resultdf =  pd.concat( [xdf,ydf], axis=1 )

    return resultdf
```

PROJECT REPORT

alldata = min_max_normalize(alldata)

alldata.tail()

from sklearn.ensemble import RandomForestClassifier as rfc

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

treatData = alldata.copy()

treatData = treatData.fillna(-1)

treatData.head()

PROJECT REPORT

```
X = treatData[[c for c in treatData.columns if c != 'pose']]

X['img'] = allFiles

X['human'] = allShots

Y = treatData['pose']



X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,

random_state=2019, shuffle=True)

#X_train, X_test, y_train, y_test = (X,X,Y,Y)



testImg = X_test.img

testHuman = X_test.human

X_train = np.array(X_train[[c for c in X_train.columns if c not in

['img','human'] ]])
```

PROJECT REPORT

```
X_test = np.array(X_test[[c for c in X_test.columns if c not in
['img','human'] ]])
```

```
print(X_train.shape)
```

```
clf = rfc(max_depth=5, n_estimators=10, random_state=2019)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
for p,img,hum in zip(y_pred,testImg,testHuman):
    fig = plt.figure(figsize = (5,5))
    a = fig.add_subplot(1, 1, 1)
```

```
    if(p==0):
        t='Sweep'
    elif(p==1):
        t='Cut'
    else:
        t='Drive'
    a.set_title(t,fontsize=20)
    resultImage = TfPoseEstimator.draw_humans(img, [hum], imgcopy=False)
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

PROJECT REPORT

Team Details:
1.DESU HARSHITH - +917893989626 - desuharshith@gmail.com
2.DOWPATI NITHISH - +916303353459 - dowpati.nithish@gmail.com
3.G.MANIKANTA - +919121198631 - manikantasairam09082002@gmail.com
4.IKKURTHI MASTAN VALLI - +916304958796 - mastanvali1601@gmail.com

College Name : Narasaraopeta engineering college

THANK YOU