A Mini Project Report on

Developing an Accurate Model for Cricket Pose Estimation

B.E. - I.T Engineering

Submitted By

Siddhesh Puranik 19104014

Niranjan Ram 19104025

Siddharth Chhoriya 20204005

Under The Guidance of

Prof. Sonal Balpande

DEPARTMENT OF INFORMATION TECHNOLOGY

A. P. SHAH INSTITUTE OF TECHNOLOGY

G.B. Road, Kasarvadavali, Thane (W), Mumbai-400615

UNIVERSITY OF MUMBAI

Academic year: 2022-23

CERTIFICATE

This to certify that the Mini Project report on Developing an Accurate Model for Cricket pose Estimation has been submitted by **Siddhesh Puranik** (19104014), **Niranjan Ram** (19104025) and **Siddharth Chhoriya** (20204005) who are a Bonafide students of A. P. Shah Institute of Technology, Thane, Mumbai, as a partial fulfillment of the requirement for the degree in **Information Technology**, during the academic year **2022-2023** in the satisfactory manner as per the curriculum laid down by University of Mumbai.

Prof. Sonal Balpande
Subject In-charge

Dr. Kiran Deshpande
Head Department of Information Technology

TABLE OF CONTENTS

Chapter 1

Introduction

Regardless of age or gender, cricket is among the top 10 sports practiced worldwide. Two teams of eleven players each compete in the team sport of cricket. Cricket is a bat-and-ball game played on a roughly oval grass field that has a pitch in the middle that is a flat strip of ground 20.12 meters (22 yards) long. There is a wicket, or set of wooden stumps, at each end of the playing field. It should be noted that, confusingly, the pitch itself is frequently referred to as the wicket. A firm, fist-sized, cork-centered leather ball is bowled by a member of the fielding team from one wicket to the other. Before the batsman from the opposite team, who uses a wooden cricket bat to protect the wicket from the ball, the ball typically bounces once before reaching him. The batsman may then run between the wickets, exchanging ends with another batsman (the "non-striker"), who has been standing in an inactive role near the bowler's wicket, to score runs. The remainder of the bowlers' team stand in various positions around the oval as fielders. In this paper, we have proposed a layered convolutional neural network for classifying various cricket poses for batting, bowling and catching. It consists of 4 convolution2D layers, 2 MaxPooling layers, 3 Dropout layers, 1 Flattening layer and 3 Dense layers.

Chapter 2

Literature Review

Sr No.	Paper Title	Findings
1.	Shot-Net: A convolutional Neural Network for Classifying Different Cricket Shots.	In this paper, they have proposed a 13 layered Convolutional Neural Network for classifying cricket shots. A CNN based model where images are served in 3 convolutional layers, 3 max pooling layers, 4 dropout layers and 2 dense layers.
2.	InceptB: A CNN Based Classification Approach for Recognizing Traditional Bengali Games.	This study trained a CNN model on a collection of images from traditional Bengali games using Google's acclaimed Inception-v3 model of TensorFlow platform and transfer learning technology.

Chapter 3

Problem Statement

Pose estimation is one of the issues that have gained many benefits from using state-of-the-art deep learning-based models. Human pose, hand and mesh estimation is a significant problem. A Human Pose Skeleton represents the orientation of a person in a graphical format. Essentially, it is a set of coordinates that can be connected to describe the pose of the person. Each coordinate in the skeleton is known as a part (or a joint, or a key point). A valid connection between two parts is known as a pair (or a limb). Note that, not all part combinations give rise to valid pairs. A wide variety of solutions have been proposed to tackle the problem. Deep Learning-based approaches have been extensively studied in recent years and used to address several computer vision problems. However, it is sometimes hard to compare these methods due to their intrinsic difference. Based on these key points we can compare various movements and postures and draw insights. With the help of this tool cricket association boards can use this for betterment of analyzing the pose of a cricketers while playing the game to identifying the different poses such as cut, sweep, drive, bowling action, fielding etc.

Chapter 4

Objectives

- To Train accurate model for Cricket Pose Detection using CNN Deep Learning Architecture.
- To collect at least 10 type of Cricket Poses Data using web scraping and Image Augmentation.
- To Create user friendly User Interface using Flask Framework.

Chapter 5

Proposed System

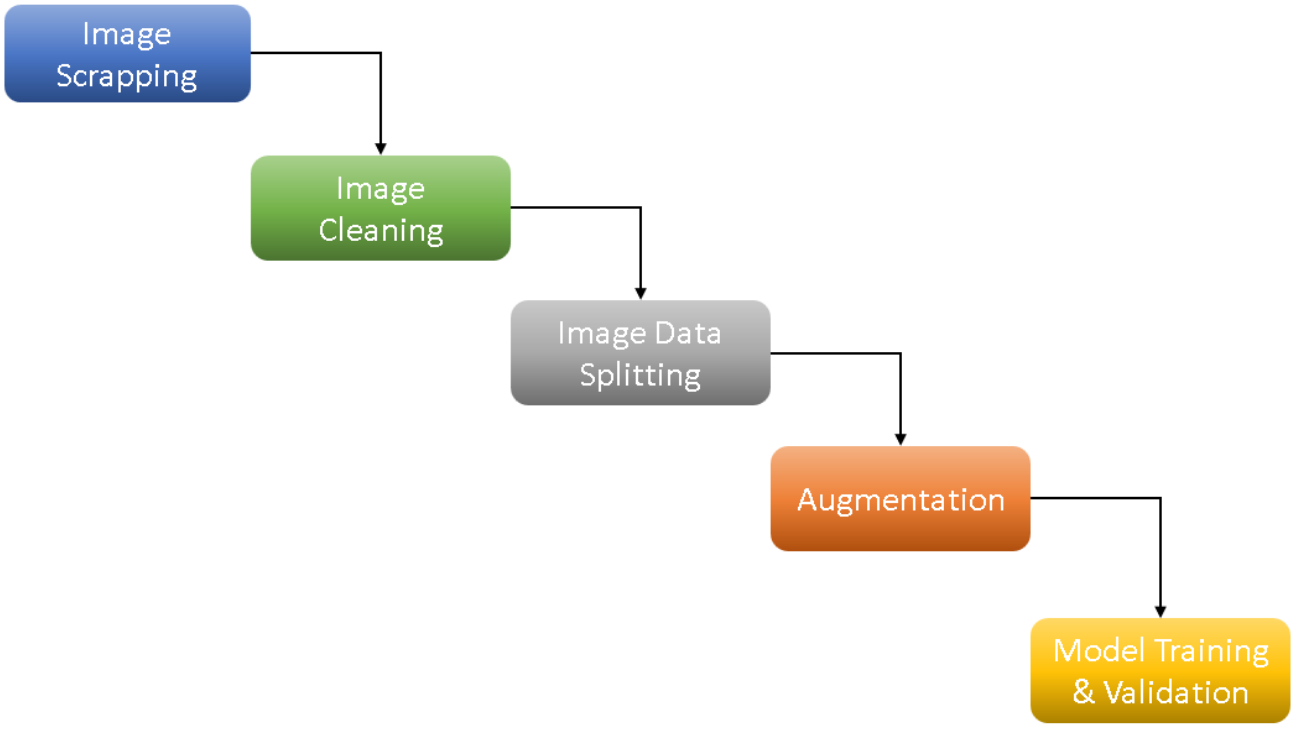

Fig 5.1

- **Image Scrapping:**

  Image scraping is a subset of the web scraping technology. While web scraping deals with all forms of web data extraction, image scraping only focuses on the media side – images, videos, audio, and so on.

- **Image Cleaning:**

  Data cleansing, also referred to as data cleaning or data scrubbing, is the process of fixing incorrect, incomplete, duplicate or otherwise erroneous data in a data set. It involves identifying data errors and then changing, updating or removing data to correct them.

- **Image Split:**

  Split and merge segmentation is an image processing technique used to segment an image. The image is successively split into quadrants based on a homogeneity criterion and similar regions are merged to create the segmented result.

- **Augmentation:**

  Image augmentation is a technique that is used to artificially expand the data-set. This is

helpful when we are given a data-set with very few data samples. In case of Deep Learning, this situation is bad as the model tends to over-fit when we train it on limited number of data samples.

- **Model training and validation:**

  In machine learning, model validation is referred to as the process where a trained model is evaluated with a testing data set. The testing data set is a separate portion of the same data set from which the training set is derived.

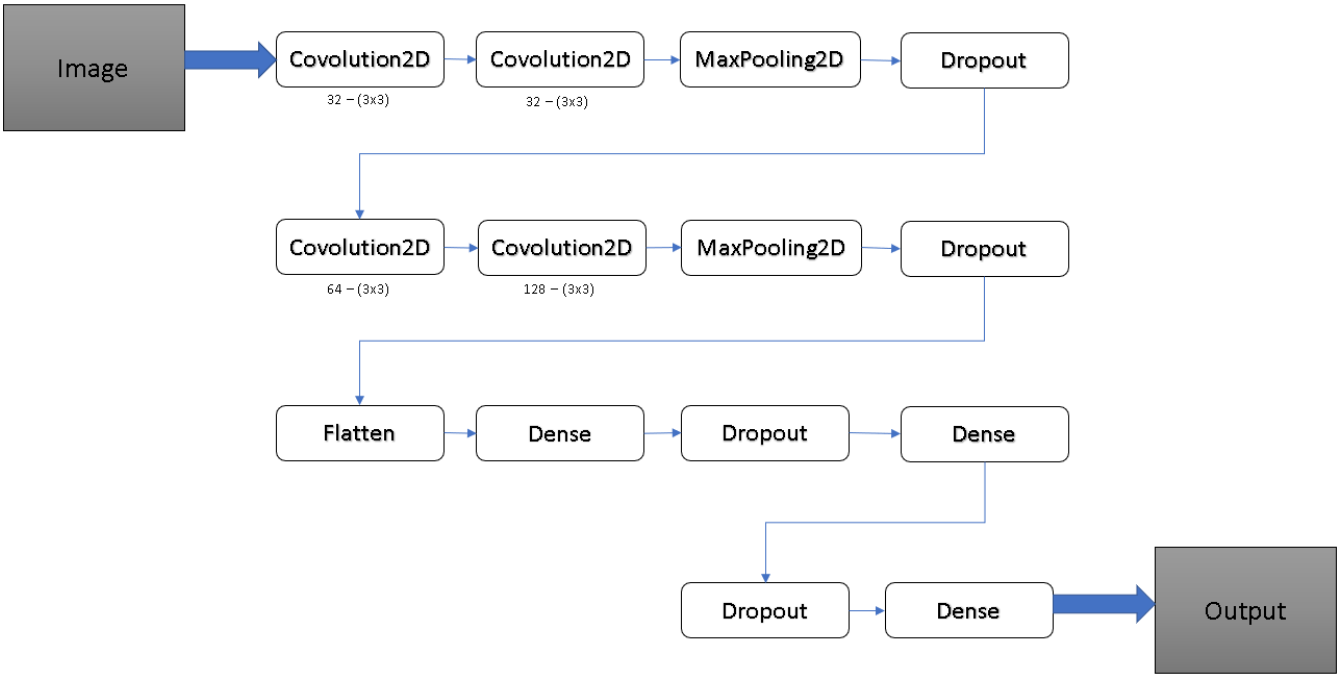

Fig 5.2

Layers:

- **Convolutional Layer**

  A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), parameters of which are to be learned throughout the training. The size of the filters is usually smaller than the actual image. Each filter convolves with the image and creates an activation map. Here we have used convolution2d that convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term.

- **Max-Pooling**

  Max pooling is a pooling operation that selects the maximum element from the region of the

feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

- **Dropout**

  The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

- **Flattening**

  Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

- **Dense**

  Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers. Working of single neuron. A layer contains multiple number of such neurons.

- **Activation Functions**

1. **LeakyReLU:**

   Leaky Rectified Linear Unit, or Leaky ReLU, is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. The slope coefficient is determined before training, i.e.it is not learnt during training. We used LeakyReLU in first two layers of covolutional2D.

2. **ReLU:**

   The usage of ReLU helps to prevent the exponential growth in the computation required to operate the neural network. If the CNN scales in size, the computational cost of adding extra ReLUs increases linearly. We used ReLU in next two layers of covolutional2D.

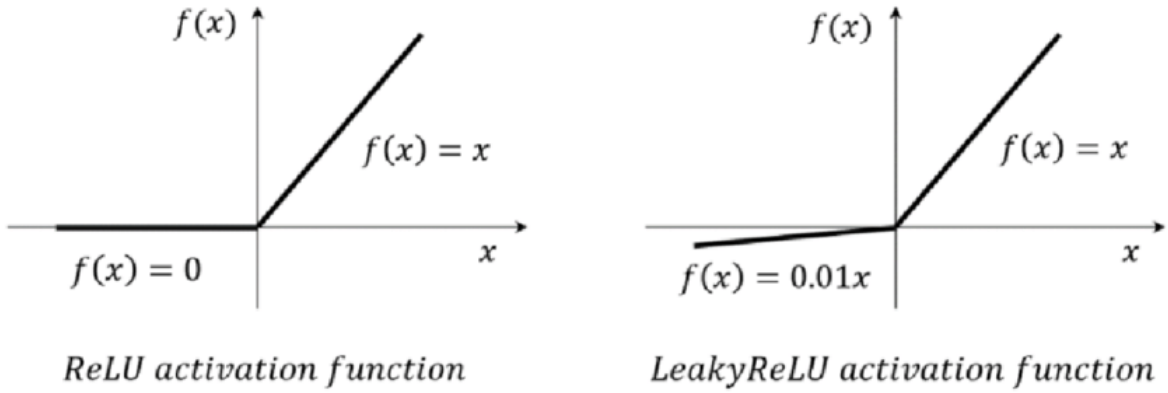
f(x)
f(x) = x
f(x) = 0
x
ReLU activation function
f(x)
f(x) = x
f(x) = 0.01x
x
LeakyReLU activation function

Fig 5.3

Chapter 6

Technological Stack

- Python:

  As Backend Programming Language.

- IBM Watson Studio:

  To create and deploy the ML model.

- CNN:

  Image Classification Neural Network Algorithm.

- Flask:

  Python-based framework to connect model and Front End.

- HTML&CSS:

  To design a User-friendly interface.

Chapter 7

Implementation

split.py

```
import splitfolders
splitfolders.fixed("images", output="data",
    seed=1337, fixed=(0, 25), oversample=False, group_prefix=None, move=False)
```

Fig 7.1 Data Splitting

agu.py

```
import Augmentor
import os
l=['bowling','catch','cover drive','dive','flick','pull shot','scoop shot','square cut','straight drive','sweep']
for i in l:
    path="data/train/"+i
    c=int(len(os.listdir(path)))
    print(c)
    p = Augmentor.Pipeline(path)
    p.flip_left_right(0.45)
    p.black_and_white(0.1)
    p.rotate(0.35, 25, 25)
    p.skew(0.15, 0.35)
    p.zoom(probability = 0.25, min_factor = 1.25, max_factor = 1.55)
    p.sample((225-c))
```

Fig 7.1 Data Augmentation

Fig 7.3 Data Feeding

```
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation=LeakyReLU(),padding='same'))
model.add(Convolution2D(32,(3,3),input_shape=(64,64,32),activation=LeakyReLU(),padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))
model.add(Convolution2D(64,(3,3),input_shape=(32,32,32),activation='relu',padding='same'))
model.add(Convolution2D(128,(3,3),input_shape=(32,32,64),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(128))
model.add(Dropout(0.3))
model.add(Dense(64))
model.add(Dropout(0.3))
model.add(Dense(10,activation='softmax'))
```

Fig 7.4 Implementation of Layers

Fig 7.5 Model Fitting

```python
from flask import Flask, flash, request, redirect, url_for, render_template
import urllib.request
import os
from werkzeug.utils import secure_filename
import numpy as np
from io import BytesIO
from tensorflow import keras
from tensorflow.keras.preprocessing import image

app = Flask(__name__)
index=['bowling','catch','cover drive','dive','flick','pull shot','scoop shot','square shot','straight drive','sweep']
UPLOAD_FOLDER = 'static/uploads/'
model=keras.models.load_model("C:/Users/USER/Desktop/Cricket/cricket.hdf5")
app.secret_key = "secret key"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024

ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/', methods=['POST'])
def upload_image():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        flash('No image selected for uploading')
        return redirect(request.url)
    if file and allowed_file(file.filename):
        filename = file.filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        #print('upload_image filename: ' + filename)

        x=image.img_to_array(image.load_img("C:/Users/USER/Desktop/Cricket/WebApp/static/uploads/"+file.filename,target_size=(64,64)))
        x=np.expand_dims(x,axis=0)
        pred=np.argmax(model.predict(x),axis=1)
        flash(index[pred[0]])
        return render_template('index.html', filename=filename)
    else:
        flash('Allowed image types are - png, jpg, jpeg, gif')
        return redirect(request.url)

@app.route('/display/<filename>')
def display_image(filename):
    #print('display_image filename: ' + filename)
    return redirect(url_for('static', filename='uploads/' + filename), code=301)

if __name__ == "__main__":
    app.run(debug=True)
```

Fig 7.6 Flask App

```
<html>
    <head>
        <title>Python Flask Upload and display image</title>
        <link rel="stylesheet"
              href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"/>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
        <link rel="stylesheet" href="static/css/main.css">
    </head>

    <body>
        <main class="container">
            <div class="bg-light p-5 rounded mt-3">
              <h1>CRICKET POSE ESTIMATION</h1>
            </div>
          </main><br><br><br><br><br>
          <div class="container">
            <div class="row">
                <h2>Select a file to upload</h2>
                <p>
                    {% with messages = get_flashed_messages() %}
                        {% if messages %}
                            <ul>
                                {% for message in messages %}<li>{{ message }}</li>{% endfor %}
                            </ul>
                        {% endif %}
                    {% endwith %}
                </p>
                {% if filename %}
                    <div>
                        <img src="{{ url_for('display_image', filename=filename) }}">
                    </div>
                {% endif %}
                <form method="post" action="/" enctype="multipart/form-data">
                    <dl>
                        <p>
                            <input type="file"
                                   name="file"
                                   class="form-control"
                                   autocomplete="off"
                                   required>
                        </p>
                    </dl>
                    <p>
                        <input type="submit" value="Submit" class="btn btn-info">
                    </p>
                </form>
            </div>
        </div>

            <script src="/docs/5.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-u1OknCvxWvY5kfmNBILK2hRnQC3Pr17a+RTT6rIHI7NnikvbZlHgTPOOmMi466C8" crossorigin="anonymous"></script>
        </body>
</html>
```

Fig 7.7 UI code

Chapter 8

Results:

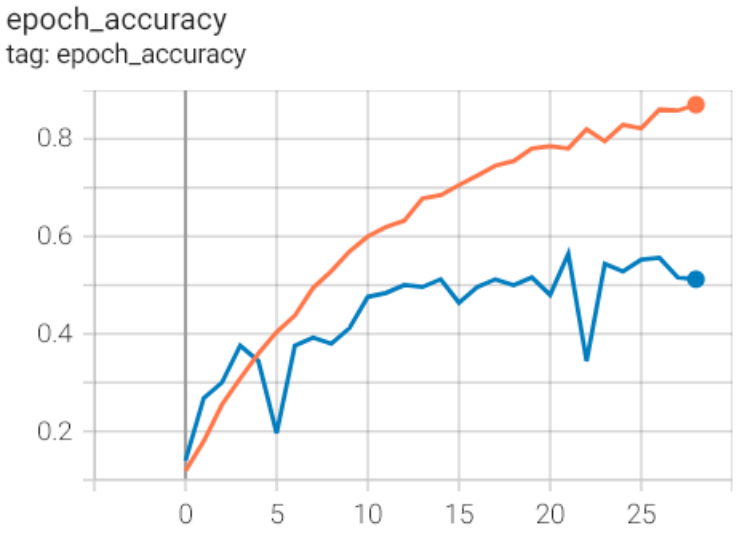

Fig 8.1

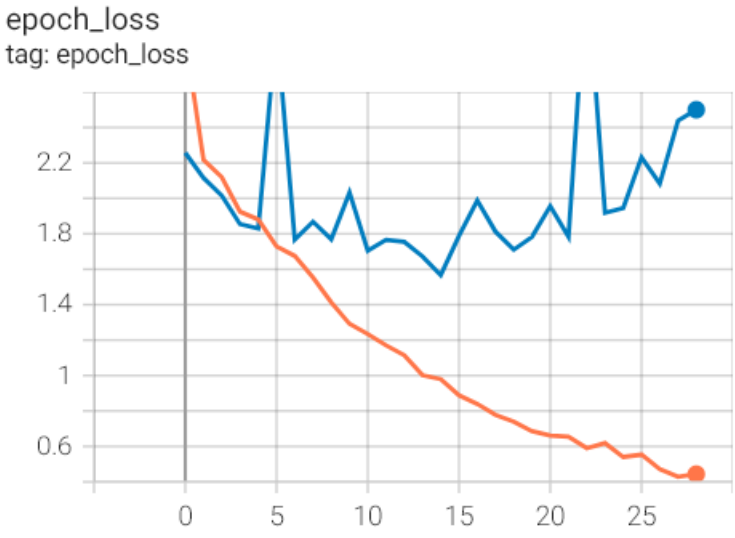

Fig 8.2

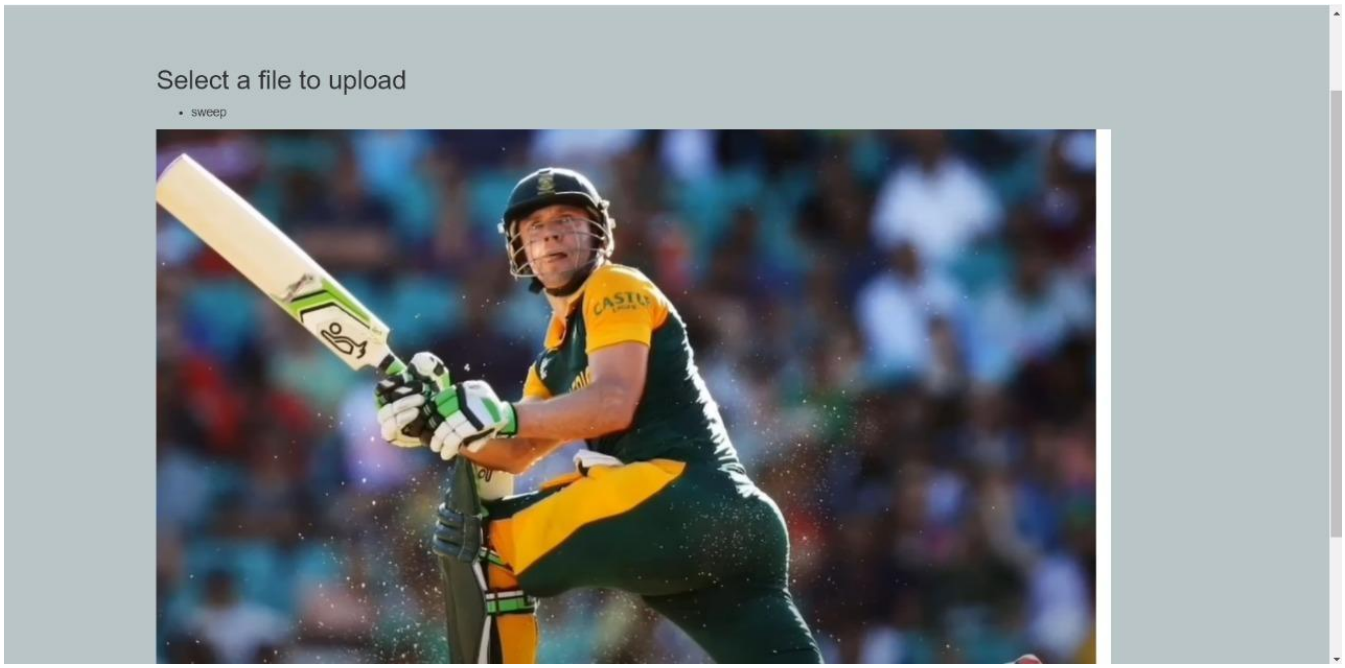
Select a file to upload
sweep

Fig 8.3

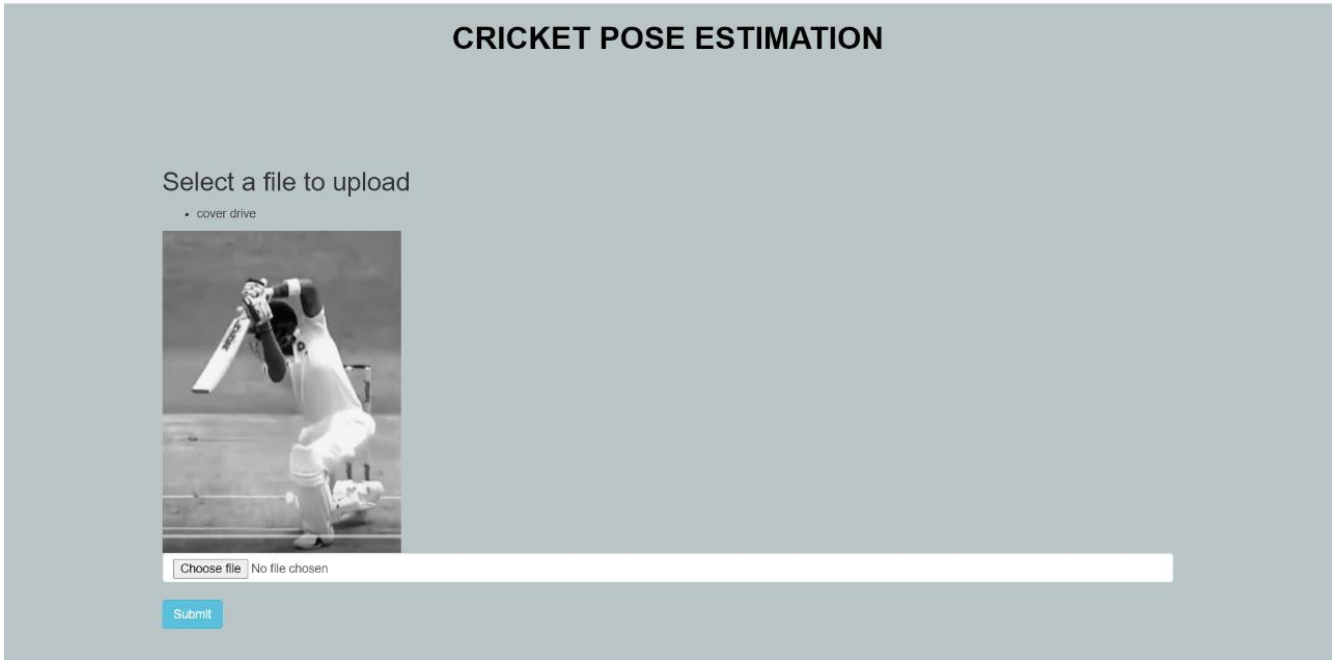
CRICKET POSE ESTIMATION
Select a file to upload
cover drive
Choose file
No file chosen
Submit

Fig 8.4

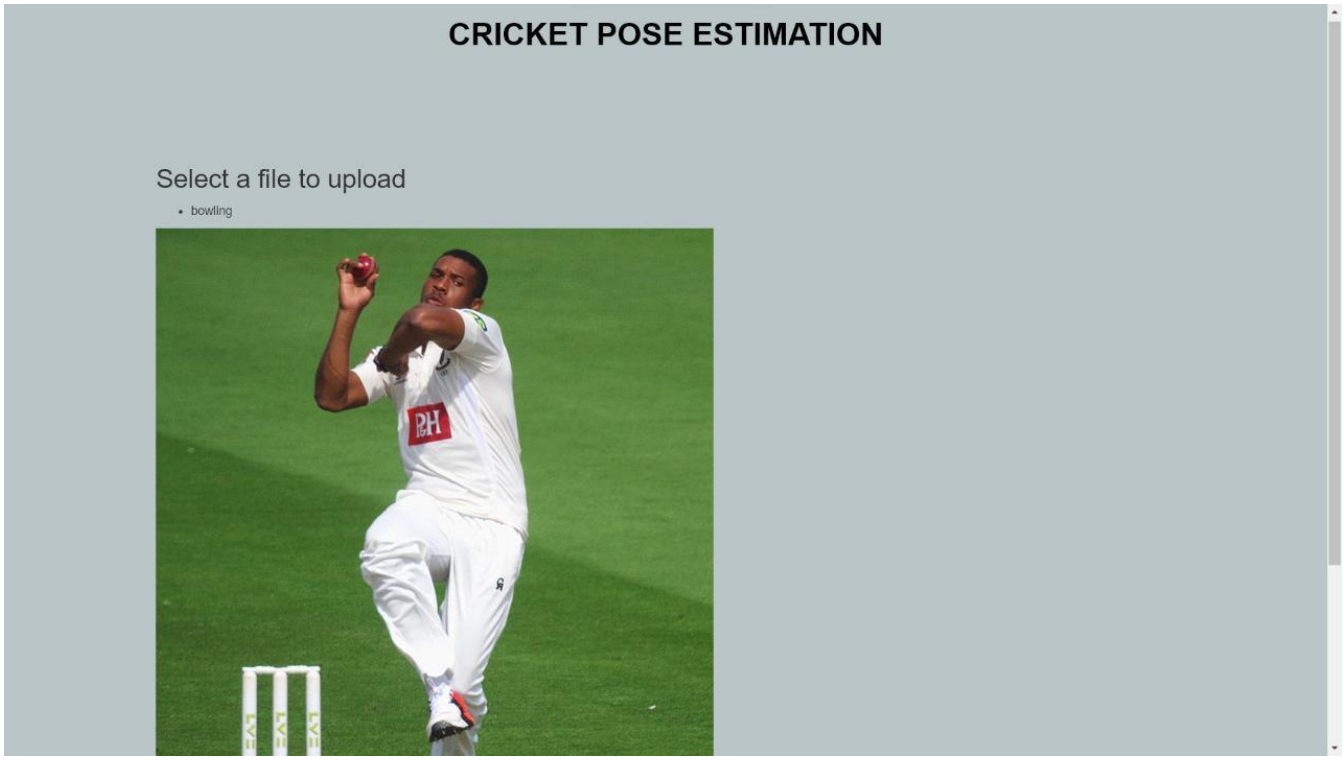

Fig 8.5

Chapter 9

Conclusion:

In this study, we present a method for classifying cricket shots using our CNN model. We used one flatten layer, three dense layers, two max polling layers, four dropout layers, and four convolution layers. To lessen over-fitting, dropout layers are used. The outcome descent with Validation Accuracy of 56% and Cross Entropy loss of 1.78. In the future, we hope that this technique will be implemented in actual applications for the benefit of the cricket game. It will work effectively for the coaching method to raise batting and bowling abilities.

Future Scope:

We have used convolution neural networks to create a model for our Shot-Net data so that our suggested method can classify various cricket shots. Making a better neural network in the future will help us achieve greater precision.

References:

[1] Foysal, Md, et al. "Shot-Net: A convolutional neural network for classifying different cricket shots." *International Conference on Recent Trends in Image Processing and Pattern Recognition*. Springer, Singapore, 2018.

[2] Islam, M.S., Foysal, F.A., Neehal, N., Karim, E., Hossain, S.A.: InceptB: a CNN based classification approach for recognizing traditional bengali games. In: ICACC2018 (2018)

[3] https://blog.jovian.ai/detecting-cricket-shots-using-pose-estimation-8e69ed12fe98?gi=e32c42dbd673