Nithin S
221IT085

IT204 Mini Project
Classical Algorithm Implementation

Research Paper Referred :

Old Research Paper

M. Han, J. Chen, L. Li and Y. Chang, "Visual hand gesture recognition with convolution neural network," 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Shanghai, China, 2016, pp. 287-291, doi: 10.1109/SNPD.2016.7515915.

Uses a Dataset of grayscaled images to create the Hand Sign Recognition ML model which has relatively less accuracy.

Recent Research Paper

S. Meshram, R. Singh, P. Pal and S. K. Singh, "Convolution Neural Network based Hand Gesture Recognition System," 2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2023, pp. 1-5, doi: 10.1109/ICAECT57570.2023.10118267.

Uses a Dataset of images which uses a hand tracking module to extract features more effectively and has more accuracy compared to the latter.

Old Research Paper's Algorithm Implementation

Tech Stack :

- Linux Environment
- Python Programming Language
- CNN Architecture
- Open CV
- Tensorflow
- Keras
- Numpy
- Google Teachable Machine

In this paper, the human hand gestures are detected and recognized using CNN classification approach. This process flow consists of hand ROI segmentation using mask image, fingers segmentation, normalization of segmented finger image and finger recognition using CNN classifier.

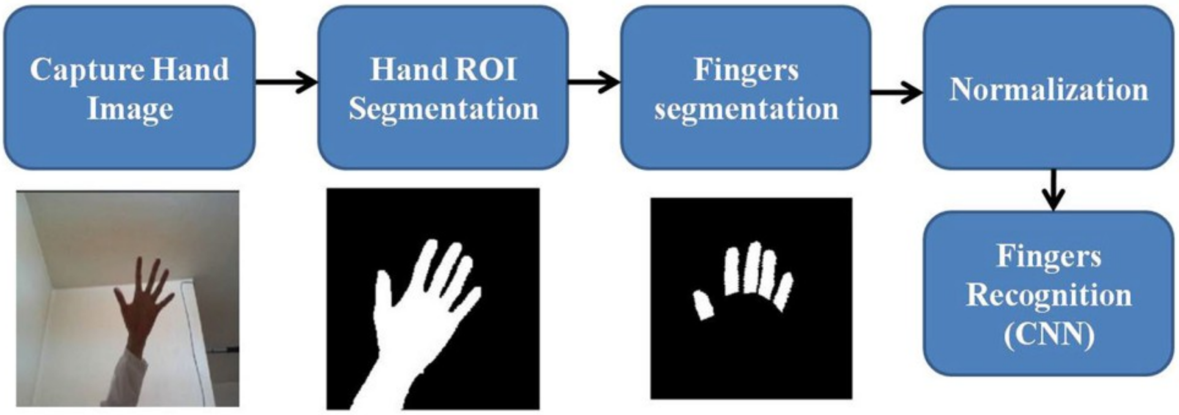


Figure shows the proposed flow of hand gesture recognition system

Algorithm

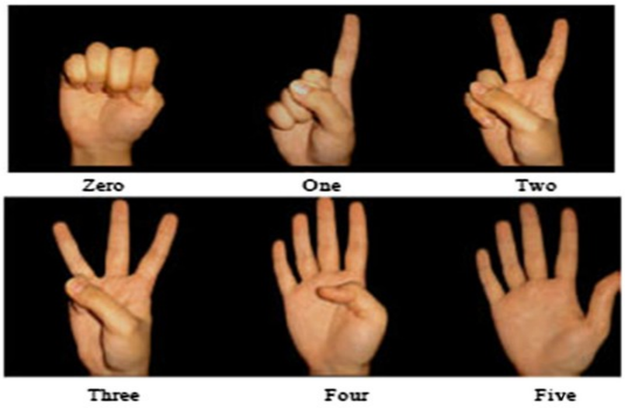
Start;

Step 1: Segment hand region ROI using hand mask images;

Step 2: Segment fingers using Connected Component Analysis algorithm;

Step 3: Classify the segmented fingers using CNN classification algorithm;

End;



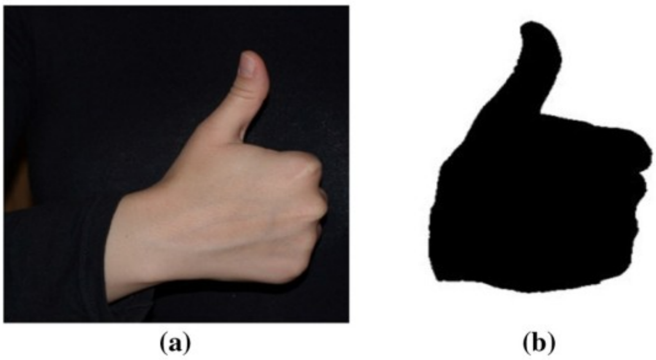
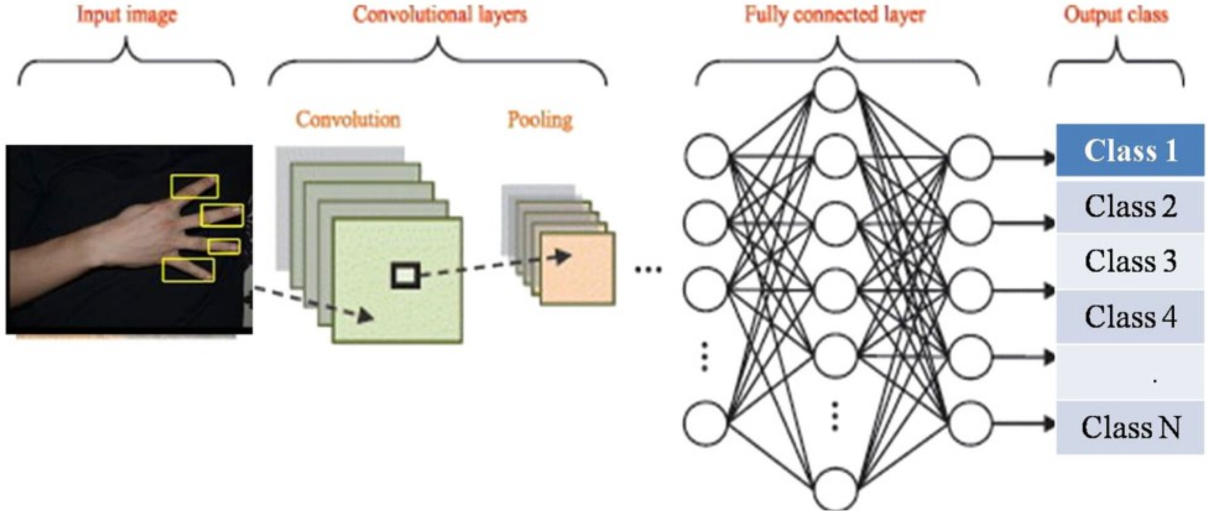
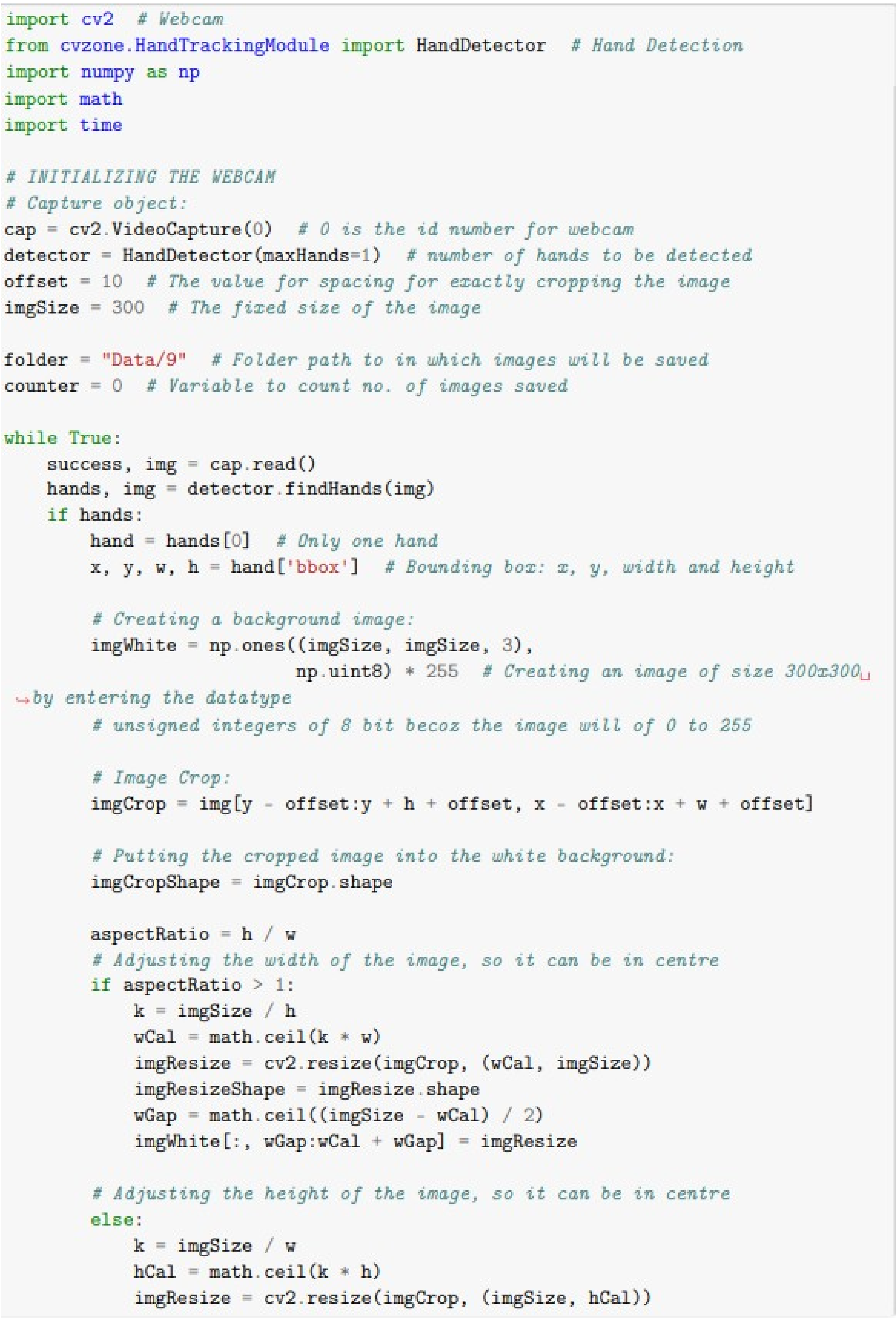


Fig. 3 **a** Hand image and **b** mask image

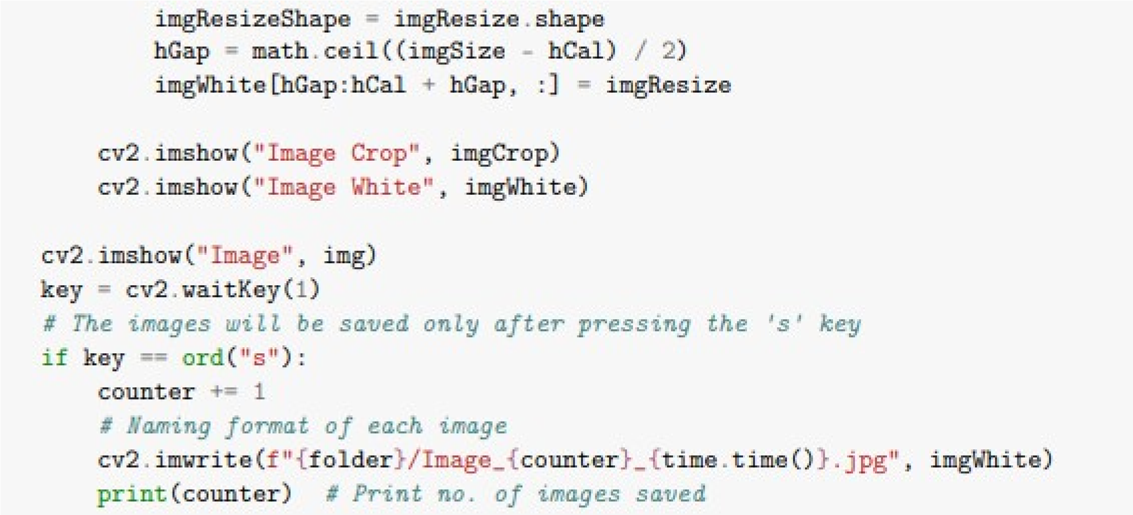


Code

dataCollection.py

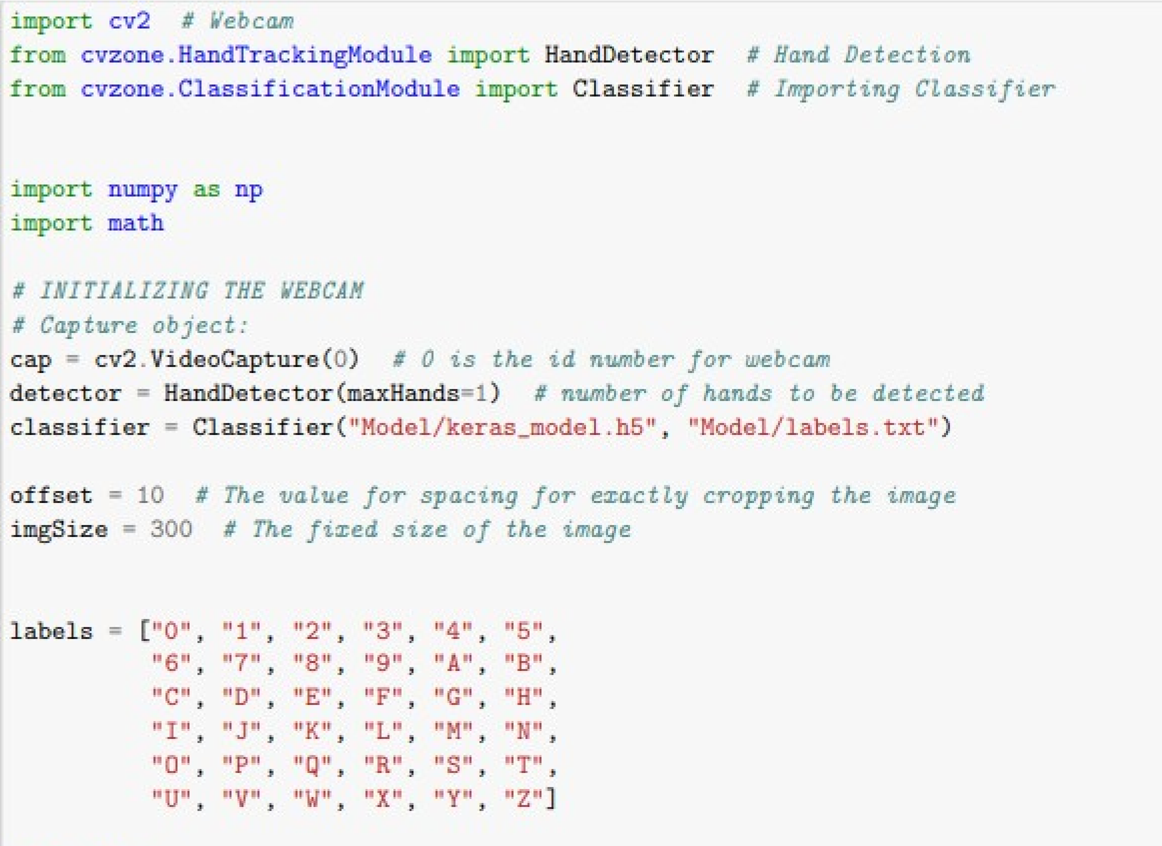


```python
import cv2  # Webcam
from cvzone.HandTrackingModule import HandDetector  # Hand Detection
import numpy as np
import math
import time

# INITIALIZING THE WEBCAM
# Capture object:
cap = cv2.VideoCapture(0)  # 0 is the id number for webcam
detector = HandDetector(maxHands=1)  # number of hands to be detected
offset = 10  # The value for spacing for exactly cropping the image
imgSize = 300  # The fixed size of the image

folder = "Data/9"  # Folder path to in which images will be saved
counter = 0  # Variable to count no. of images saved

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]  # Only one hand
        x, y, w, h = hand['bbox']  # Bounding box: x, y, width and height

        # Creating a background image:
        imgWhite = np.ones((imgSize, imgSize, 3),
                           np.uint8) * 255  # Creating an image of size 300x300␣
 ↪by entering the datatype
        # unsigned integers of 8 bit becoz the image will of 0 to 255

        # Image Crop:
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        # Putting the cropped image into the white background:
        imgCropShape = imgCrop.shape

        aspectRatio = h / w
        # Adjusting the width of the image, so it can be in centre
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize

        # Adjusting the height of the image, so it can be in centre
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
```



```
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize

        cv2.imshow("Image Crop", imgCrop)
        cv2.imshow("Image White", imgWhite)

    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
    # The images will be saved only after pressing the 's' key
    if key == ord("s"):
        counter += 1
        # Naming format of each image
        cv2.imwrite(f"{folder}/Image_{counter}_{time.time()}.jpg", imgWhite)
        print(counter)  # Print no. of images saved
```

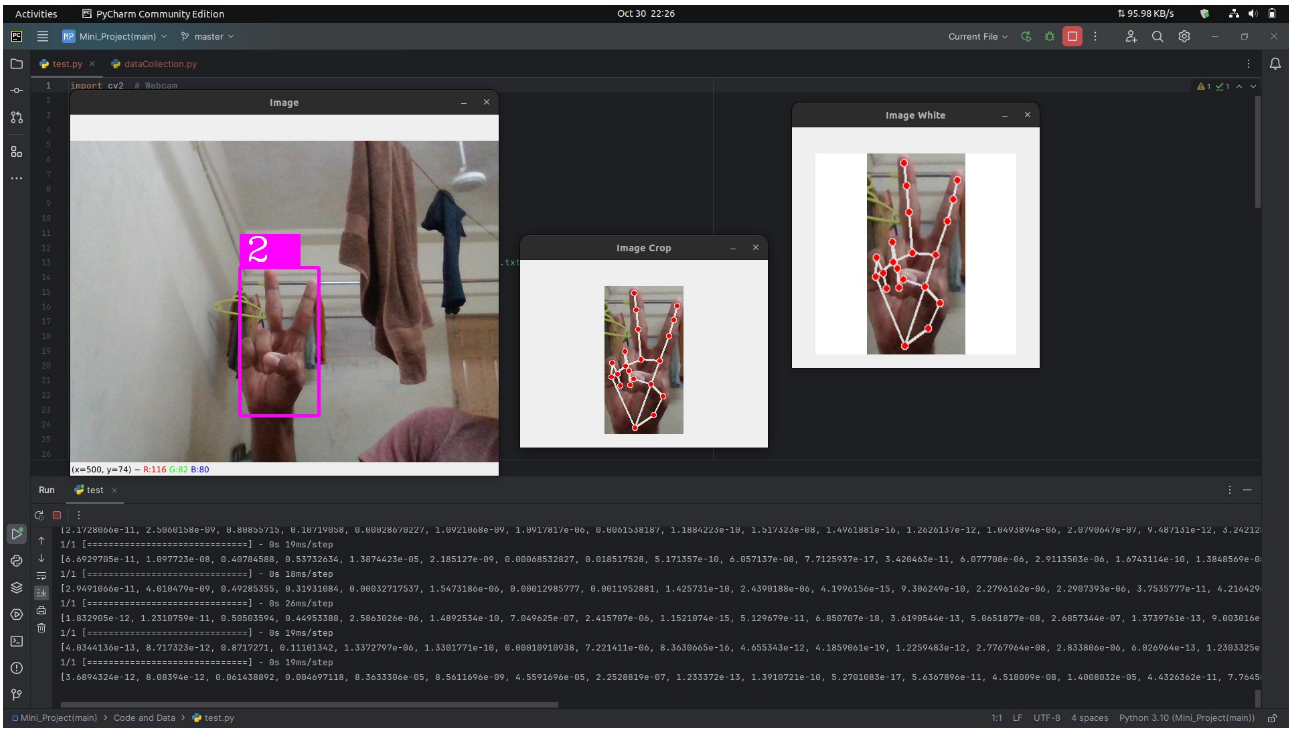
test.py



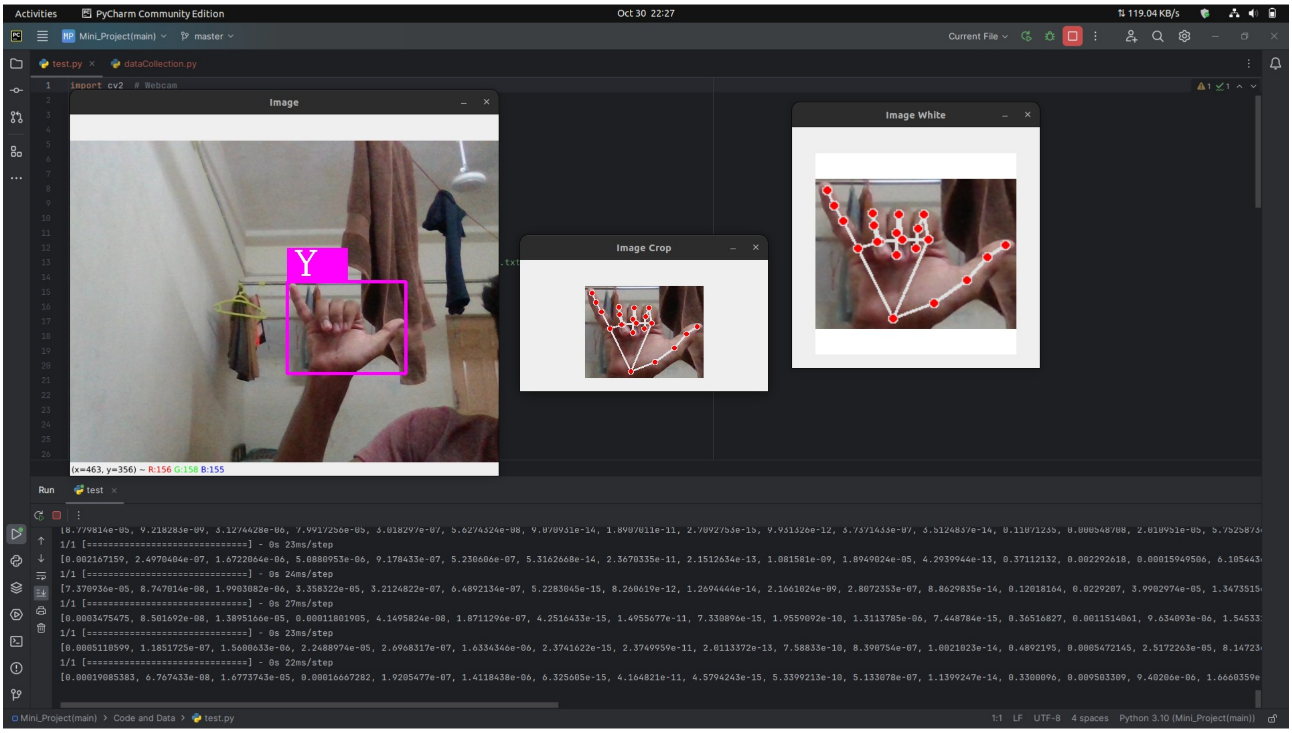
```
import cv2  # Webcam
from cvzone.HandTrackingModule import HandDetector  # Hand Detection
from cvzone.ClassificationModule import Classifier  # Importing Classifier


import numpy as np
import math

# INITIALIZING THE WEBCAM
# Capture object:
cap = cv2.VideoCapture(0)  # 0 is the id number for webcam
detector = HandDetector(maxHands=1)  # number of hands to be detected
classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")

offset = 10  # The value for spacing for exactly cropping the image
imgSize = 300  # The fixed size of the image


labels = ["0", "1", "2", "3", "4", "5",
          "6", "7", "8", "9", "A", "B",
          "C", "D", "E", "F", "G", "H",
          "I", "J", "K", "L", "M", "N",
          "O", "P", "Q", "R", "S", "T",
          "U", "V", "W", "X", "Y", "Z"]
```

```
while True:
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]  # Only one hand
        x, y, w, h = hand['bbox']  # Bounding box: x, y, width and height

        # Creating a background image:
        imgWhite = np.ones((imgSize, imgSize, 3),
                           np.uint8) * 255  # Creating an image of size 300x300␣
↪by entering the datatype
        # unsigned integers of 8 bit becoz the image will of 0 to 255

        # Image Crop:
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        # Putting the cropped image into the white background:
        imgCropShape = imgCrop.shape

        aspectRatio = h / w
        # Adjusting the width of the image, so it can be in centre
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)

        # Adjusting the height of the image, so it can be in centre
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)

        cv2.rectangle(imgOutput, (x-offset, y-offset-50), (x-offset+90,␣
↪y-offset-50+50), (255, 0, 255), cv2.FILLED)
        cv2.putText(imgOutput, labels[index], (x, y-20), cv2.
↪FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
        cv2.rectangle(imgOutput, (x-offset, y-offset), (x+w+offset, y+h+offset),␣
↪(255, 0, 255), 4)

        cv2.imshow("Image Crop", imgCrop)
        cv2.imshow("Image White", imgWhite)

    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1)
```

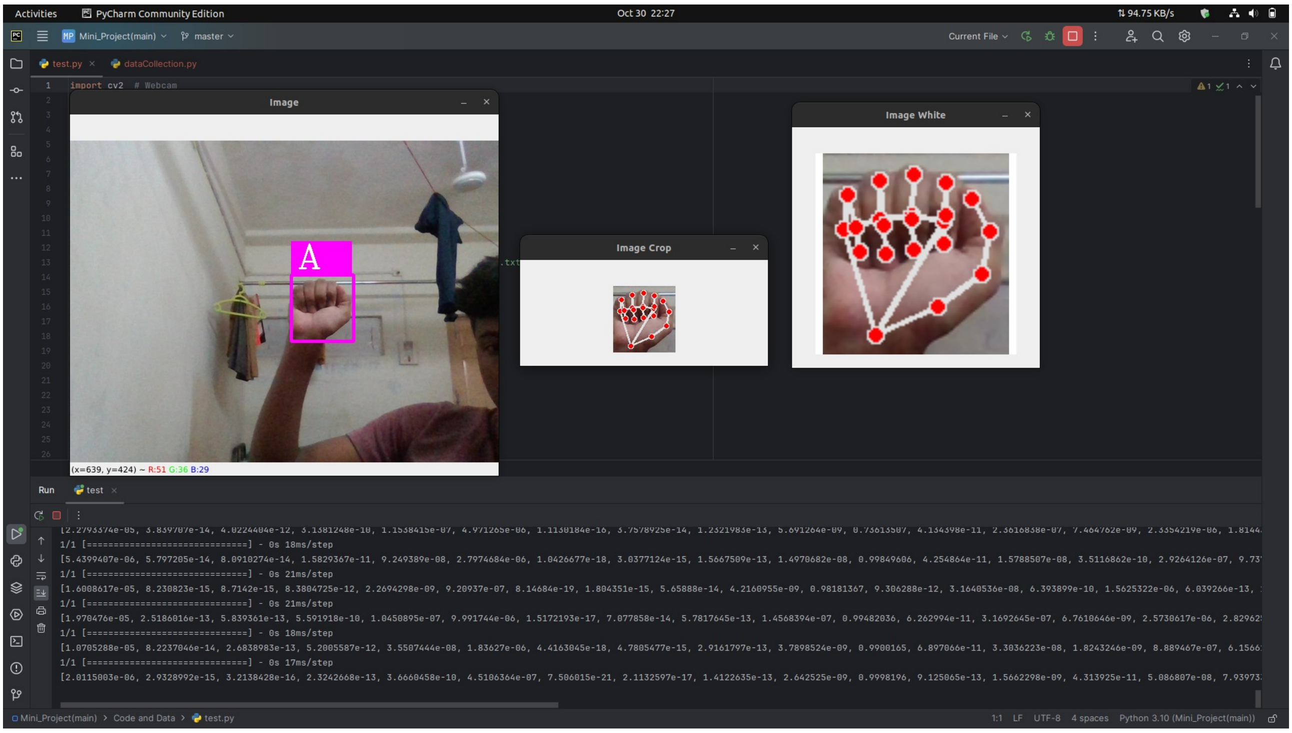
Results


Activities
PyCharm Community Edition
Oct 30 22:26
95.98 KB/s
Mini_Project(main)
master
Current File
test.py
dataCollection.py
import cv2 # Webcam
Image
2
(x=500, y=74) ~ R:116 G:82 B:80
Image Crop
Image White
Run
test
Mini_Project(main) > Code and Data > test.py
1:1 LF UTF-8 4 spaces Python 3.10 (Mini_Project(main))

```
[2.1728066e-11, 2.5060158e-09, 0.80855715, 0.10719058, 0.00028670227, 1.0921068e-09, 1.0917817e-06, 0.0061538187, 1.1884223e-10, 1.517323e-08, 1.4961881e-16, 1.2626137e-12, 1.0493894e-06, 2.0790647e-07, 9.487131e-12, 3.24212
1/1 [==============================] - 0s 19ms/step
[6.6929705e-11, 1.097723e-08, 0.40784588, 0.53732634, 1.3874423e-05, 2.185127e-09, 0.00068532827, 0.018517528, 5.171357e-10, 6.057137e-08, 7.7125937e-17, 3.420463e-11, 6.077708e-06, 2.9113503e-06, 1.6743114e-10, 1.3848569e-0
1/1 [==============================] - 0s 18ms/step
[2.9491066e-11, 4.010479e-09, 0.49285355, 0.31931084, 0.00032717537, 1.5473186e-06, 0.00012985777, 0.0011952881, 1.425731e-10, 2.4390188e-06, 4.1996156e-15, 9.306249e-10, 2.2796162e-06, 2.2907393e-06, 3.7535777e-11, 4.216429
1/1 [==============================] - 0s 26ms/step
[1.832905e-12, 1.2310759e-11, 0.50503594, 0.44953388, 2.5863026e-06, 1.4892534e-10, 7.049625e-07, 2.415707e-06, 1.1521074e-15, 5.129679e-11, 6.850707e-18, 3.6190544e-13, 5.0651877e-08, 2.6857344e-07, 1.3739761e-13, 9.003016e
1/1 [==============================] - 0s 19ms/step
[4.0344136e-13, 8.717323e-12, 0.8717271, 0.11101342, 1.3372797e-06, 1.3301771e-10, 0.00010910938, 7.221411e-06, 8.3630665e-16, 4.655343e-12, 4.1859061e-19, 1.2259483e-12, 2.7767964e-08, 2.833806e-06, 6.026964e-13, 1.2303325e
1/1 [==============================] - 0s 19ms/step
[3.6894324e-12, 8.08394e-12, 0.061438892, 0.004697118, 8.3633306e-05, 8.5611696e-09, 4.5591696e-05, 2.2528819e-07, 1.233372e-13, 1.3910721e-10, 5.2701083e-17, 5.6367896e-11, 4.518009e-08, 1.4008032e-05, 4.4326362e-11, 7.7645
```


Activities
PyCharm Community Edition
Oct 30 22:27
119.04 KB/s
Mini_Project(main)
master
Current File
test.py
dataCollection.py
import cv2 # Webcam
Image
Y
(x=463, y=356) ~ R:156 G:158 B:155
Image Crop
Image White
Run
test
Mini_Project(main) > Code and Data > test.py
1:1 LF UTF-8 4 spaces Python 3.10 (Mini_Project(main))

```
[8.779814e-05, 9.218283e-09, 3.1274428e-06, 7.9917256e-05, 3.018297e-07, 5.6274324e-08, 9.070931e-14, 1.8907011e-11, 2.7092753e-15, 9.931326e-12, 3.7371433e-07, 3.5124837e-14, 0.11071235, 0.000548708, 2.010951e-05, 5.752587
1/1 [==============================] - 0s 23ms/step
[0.002167159, 2.4970404e-07, 1.6722064e-06, 5.0880953e-06, 9.178433e-07, 5.230606e-07, 5.3162668e-14, 2.3670335e-11, 2.1512634e-13, 1.081581e-09, 1.8949024e-05, 4.2939944e-13, 0.37112132, 0.002292618, 0.00015949506, 6.105443
1/1 [==============================] - 0s 24ms/step
[7.370936e-05, 8.747014e-08, 1.9903082e-06, 3.358322e-05, 3.2124822e-07, 6.4892134e-07, 5.2283045e-15, 8.260619e-12, 1.2694444e-14, 2.1661024e-09, 2.8072353e-07, 8.8629835e-14, 0.12018164, 0.0229207, 3.9902974e-05, 1.347351
1/1 [==============================] - 0s 27ms/step
[0.0003475475, 8.501692e-08, 1.3895166e-05, 0.00011801905, 4.1495824e-08, 1.8711296e-07, 4.2516433e-15, 1.4955677e-11, 7.330896e-15, 1.9559092e-10, 1.3113785e-06, 7.448784e-15, 0.36516827, 0.0011514061, 9.634093e-06, 1.54533
1/1 [==============================] - 0s 23ms/step
[0.0005110599, 1.1851725e-07, 1.5600633e-06, 2.2488974e-05, 2.6968317e-07, 1.6334346e-06, 2.3741622e-15, 2.3749959e-11, 2.0113372e-13, 7.58833e-10, 8.390754e-07, 1.0021023e-14, 0.4892195, 0.0005472145, 2.5172263e-05, 8.14723
1/1 [==============================] - 0s 22ms/step
[0.00019085383, 6.767433e-08, 1.6773743e-05, 0.00016667282, 1.9205477e-07, 1.4118438e-06, 6.325605e-15, 4.164821e-11, 4.5794243e-15, 5.3399213e-10, 5.133078e-07, 1.1399247e-14, 0.3300096, 0.009503309, 9.40206e-06, 1.6660359e
```


Activities
PyCharm Community Edition
Oct 30 22:27
94.75 KB/s
Mini_Project(main)
master
Current File
test.py
dataCollection.py
Image
A
(x=639, y=424) ~ R:51 G:36 B:29
Image Crop
Image White
Run
test
Mini_Project(main) > Code and Data > test.py
1:1 LF UTF-8 4 spaces Python 3.10 (Mini_Project(main))

```
import cv2  # Webcam
```

```
[2.2793374e-05, 3.839707e-14, 4.0224404e-12, 3.1381248e-10, 1.1538415e-07, 4.971265e-06, 1.1130184e-16, 3.7578925e-14, 1.2321983e-13, 5.691264e-09, 0.73613507, 4.134398e-11, 2.3616838e-07, 7.464762e-09, 2.3354219e-06, 1.8144
1/1 [==============================] - 0s 18ms/step
[5.4399407e-06, 5.797205e-14, 8.0910274e-14, 1.5829367e-11, 9.249389e-08, 2.7974684e-06, 1.0426677e-18, 3.0377124e-15, 1.5667509e-13, 1.4970682e-08, 0.99849606, 4.254864e-11, 1.5788507e-08, 3.5116862e-10, 2.9264126e-07, 9.73
1/1 [==============================] - 0s 21ms/step
[1.6008617e-05, 8.230823e-15, 8.7142e-15, 8.3804725e-12, 2.2694298e-09, 9.20937e-07, 8.14684e-19, 1.804351e-15, 5.65888e-14, 4.2160955e-09, 0.98181367, 9.306288e-12, 3.1640536e-08, 6.393899e-10, 1.5625322e-06, 6.039266e-13,
1/1 [==============================] - 0s 21ms/step
[1.970476e-05, 2.5186016e-13, 5.839361e-13, 5.591918e-10, 1.0450895e-07, 9.991744e-06, 1.5172193e-17, 7.077858e-14, 5.7817645e-13, 1.4568394e-07, 0.99482036, 6.262994e-11, 3.1692645e-07, 6.7610646e-09, 2.5730617e-06, 2.82962
1/1 [==============================] - 0s 18ms/step
[1.0705288e-05, 8.2237046e-14, 2.6838983e-13, 5.2005587e-12, 3.5507444e-08, 1.83627e-06, 4.4163045e-18, 4.7805477e-15, 2.9161797e-13, 3.7898524e-09, 0.9900165, 6.897066e-11, 3.3036223e-08, 1.8243246e-09, 8.889467e-07, 6.1566
1/1 [==============================] - 0s 17ms/step
[2.0115003e-06, 2.9328992e-15, 3.2138428e-16, 2.3242668e-13, 3.6660458e-10, 4.5106364e-07, 7.506015e-21, 2.1132597e-17, 1.4122635e-13, 2.642525e-09, 0.9998196, 9.125065e-13, 1.5662298e-09, 4.313925e-11, 5.086807e-08, 7.93973
```

Conclusion

In this paper, deep learning convolutional neural network based hand gesture detection and recognition methodology
is proposed. This proposed method segments the finger tips
from the hand gesture image, and then, this finger tips are
given as input to the CNN classifier.

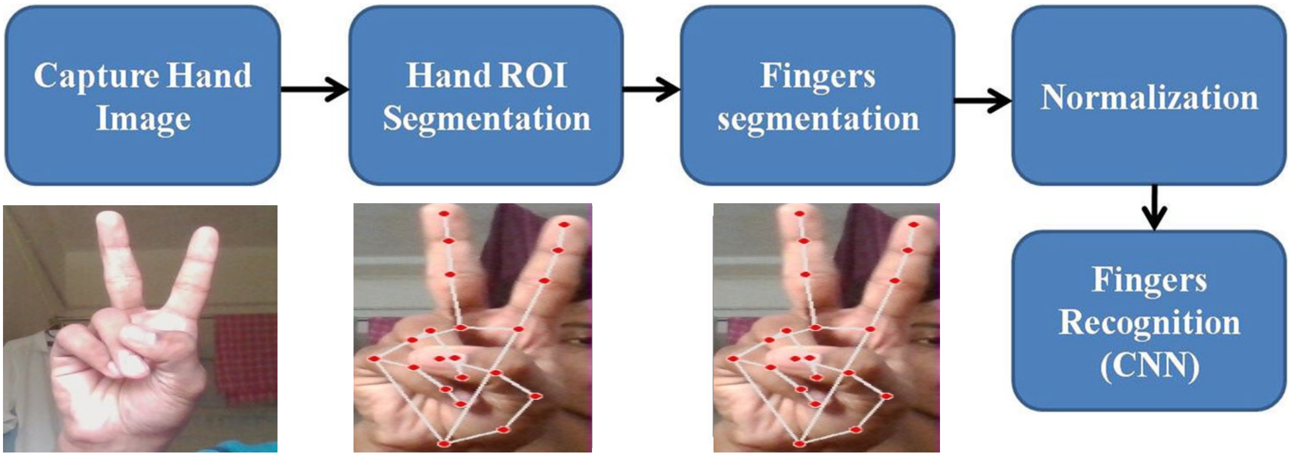
The CNN classification approach trains and classifies the test hand gesture image which is obtained from open access image dataset. The performance of the proposed hand gesturedetection and recognition methodology is analyzed in terms of sensitivity, specificity, accuracy and recognition rate. The proposed hand gesture detection and recognition methodology using CNN classification approach stated in this paper achieves 88.1% of sensitivity, 83.4% of specificity, 86.2% of accuracy and 86.2% of recognition rate.

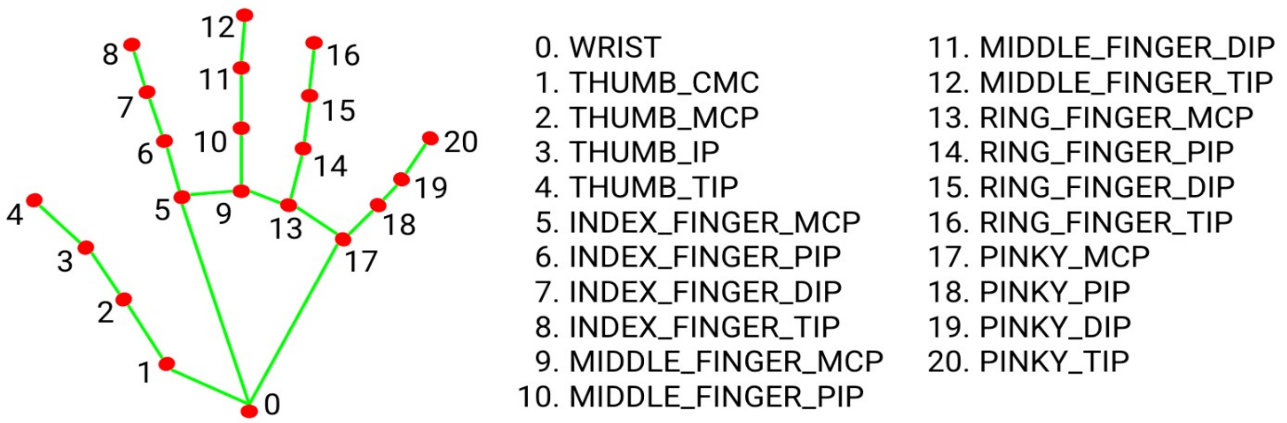
Resent Research Paper's Algorithm Implementation

Tech Stack :

- Linux Environment
- Python Programming Language
- CNN Architecture
- Open CV
- Tensorflow
- Keras
- Numpy
- Google Teachable Machine

In this paper, the human hand gestures are detected and recognized using CNN classification approach. This process flow consists of hand ROI segmentation using mask image, fingers segmentation, normalization of segmented finger image and finger recognition using CNN classifier.





0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Algorithm

Start;

Step 1: Segment hand region ROI using hand tracking module;

Step 2: Segment fingers using Connected Component Analysis algorithm;

Step 3: Classify the segmented fingers using CNN classification algorithm;

End;

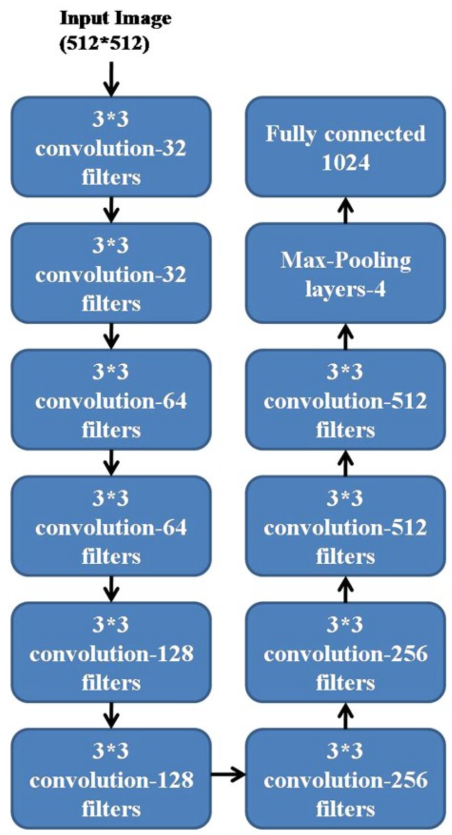

Input Image
(512*512)
3*3 convolution-32 filters
3*3 convolution-32 filters
3*3 convolution-64 filters
3*3 convolution-64 filters
3*3 convolution-128 filters
3*3 convolution-128 filters
3*3 convolution-256 filters
3*3 convolution-256 filters
3*3 convolution-512 filters
3*3 convolution-512 filters
Max-Pooling layers-4
Fully connected 1024

Fig. 6 Developed CNN architecture used in hand gesture recognition

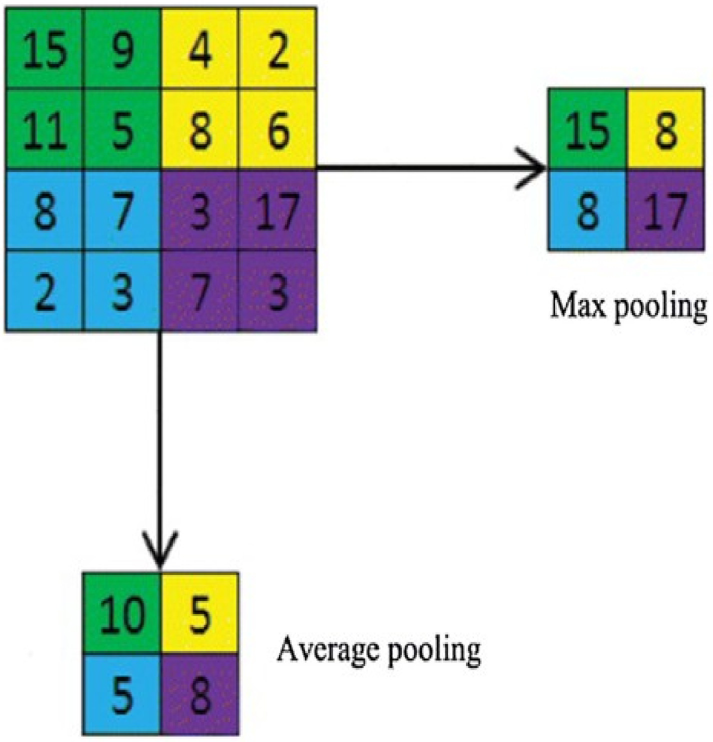
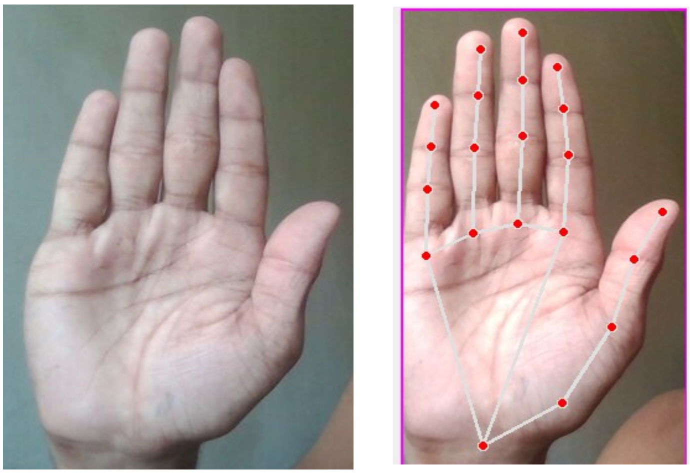


Fig. 8 Illustrations of average and max pooling



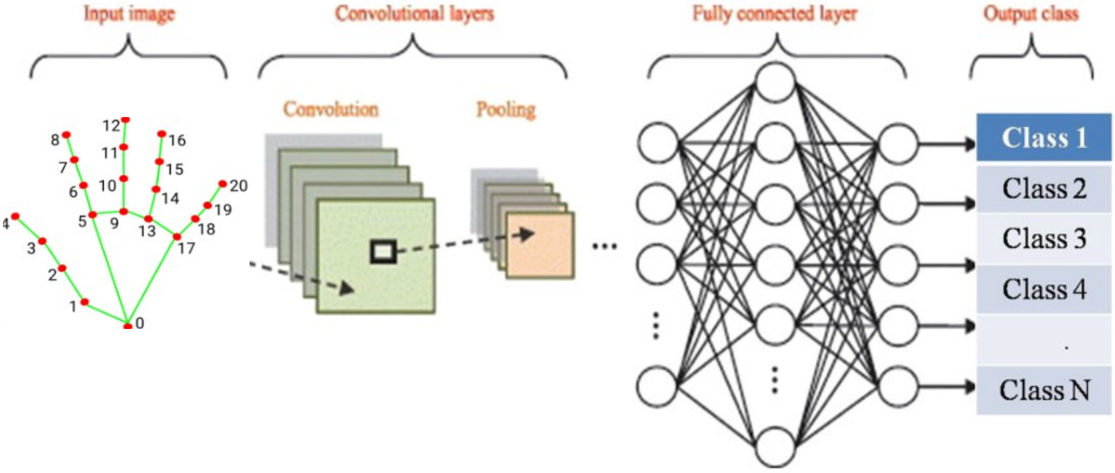
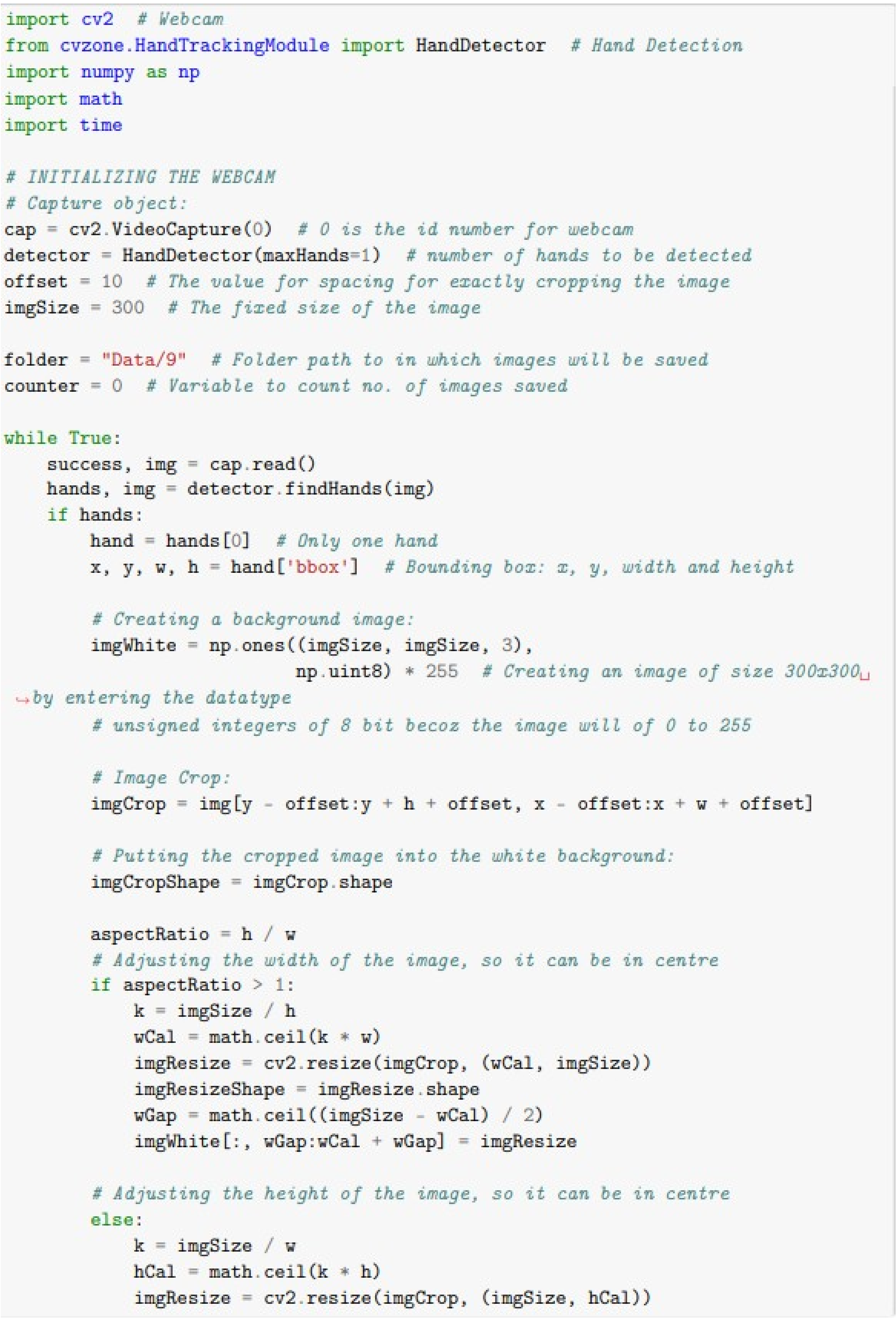


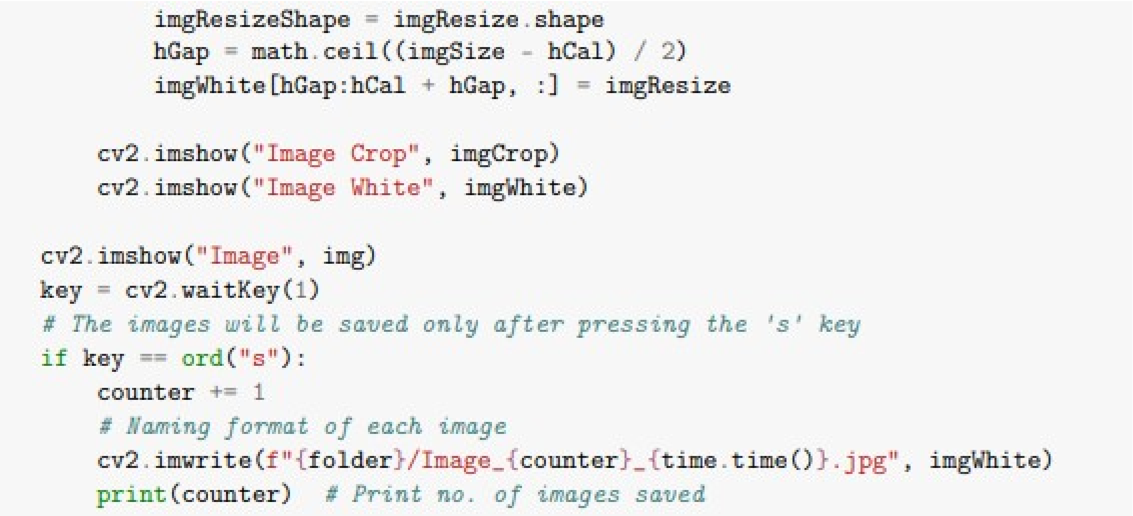
Fig. 7 Internal architecture of proposed CNN classifier for hand gesture recognition

Code

dataCollection.py

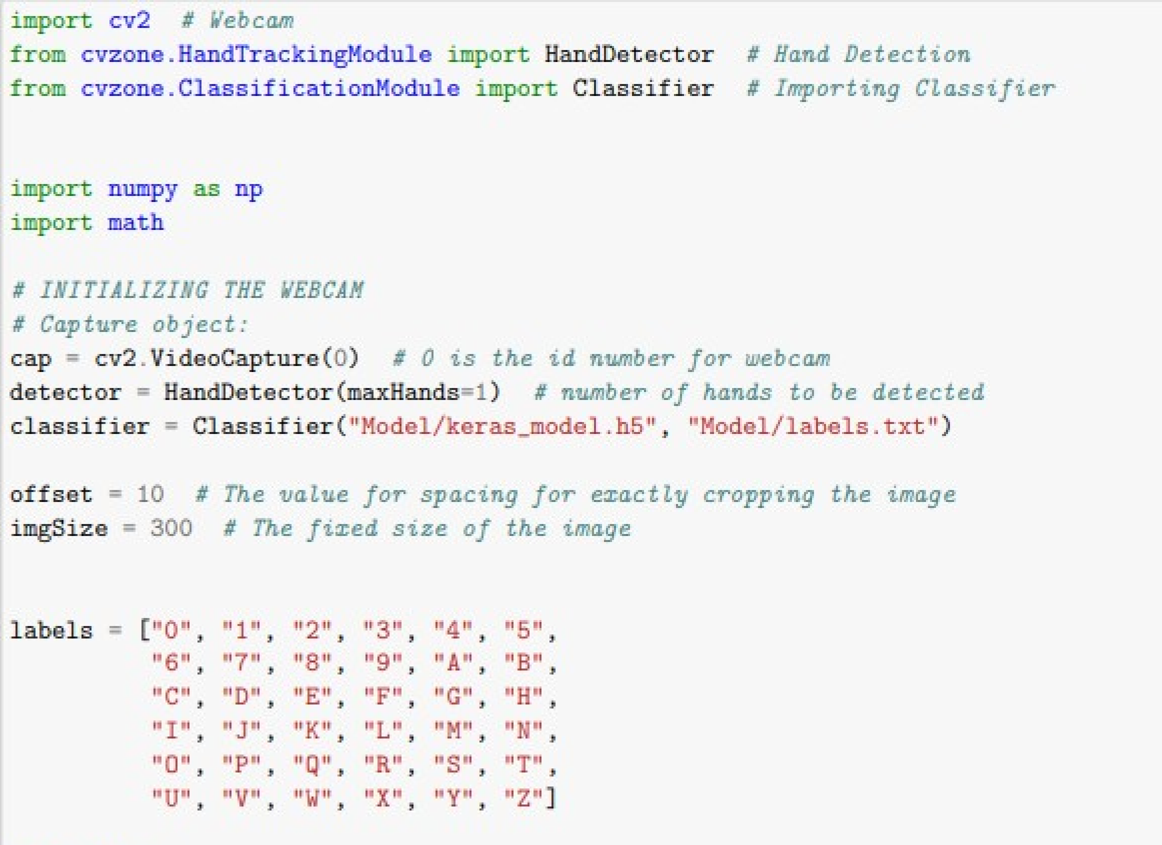


```
import cv2  # Webcam
from cvzone.HandTrackingModule import HandDetector  # Hand Detection
import numpy as np
import math
import time

# INITIALIZING THE WEBCAM
# Capture object:
cap = cv2.VideoCapture(0)  # 0 is the id number for webcam
detector = HandDetector(maxHands=1)  # number of hands to be detected
offset = 10  # The value for spacing for exactly cropping the image
imgSize = 300  # The fixed size of the image

folder = "Data/9"  # Folder path to in which images will be saved
counter = 0  # Variable to count no. of images saved

while True:
    success, img = cap.read()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]  # Only one hand
        x, y, w, h = hand['bbox']  # Bounding box: x, y, width and height

        # Creating a background image:
        imgWhite = np.ones((imgSize, imgSize, 3),
                           np.uint8) * 255  # Creating an image of size 300x300␣
↪by entering the datatype
        # unsigned integers of 8 bit becoz the image will of 0 to 255

        # Image Crop:
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        # Putting the cropped image into the white background:
        imgCropShape = imgCrop.shape

        aspectRatio = h / w
        # Adjusting the width of the image, so it can be in centre
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize

        # Adjusting the height of the image, so it can be in centre
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
```

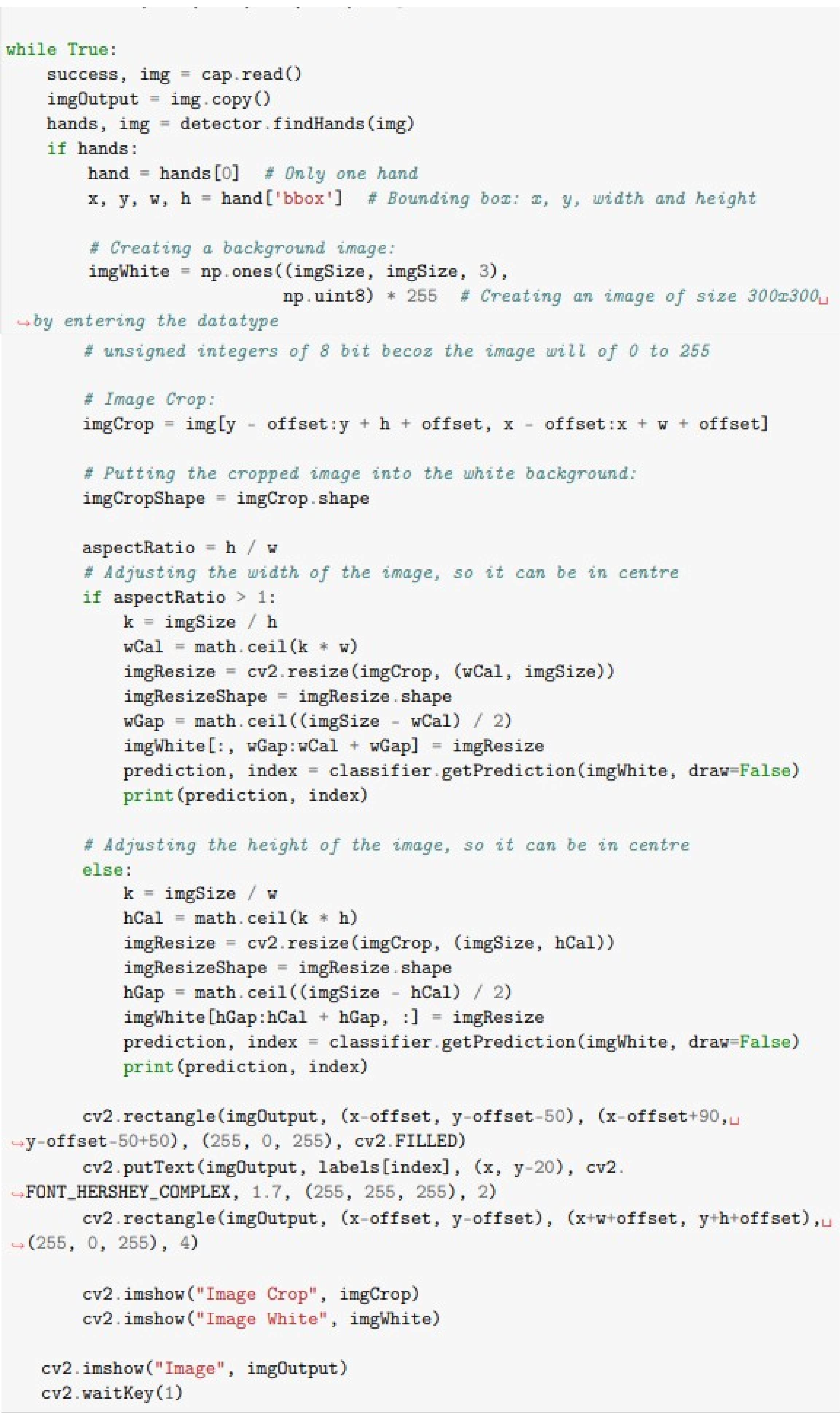


```
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize

        cv2.imshow("Image Crop", imgCrop)
        cv2.imshow("Image White", imgWhite)

    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
    # The images will be saved only after pressing the 's' key
    if key == ord("s"):
        counter += 1
        # Naming format of each image
        cv2.imwrite(f"{folder}/Image_{counter}_{time.time()}.jpg", imgWhite)
        print(counter)  # Print no. of images saved
```

test.py

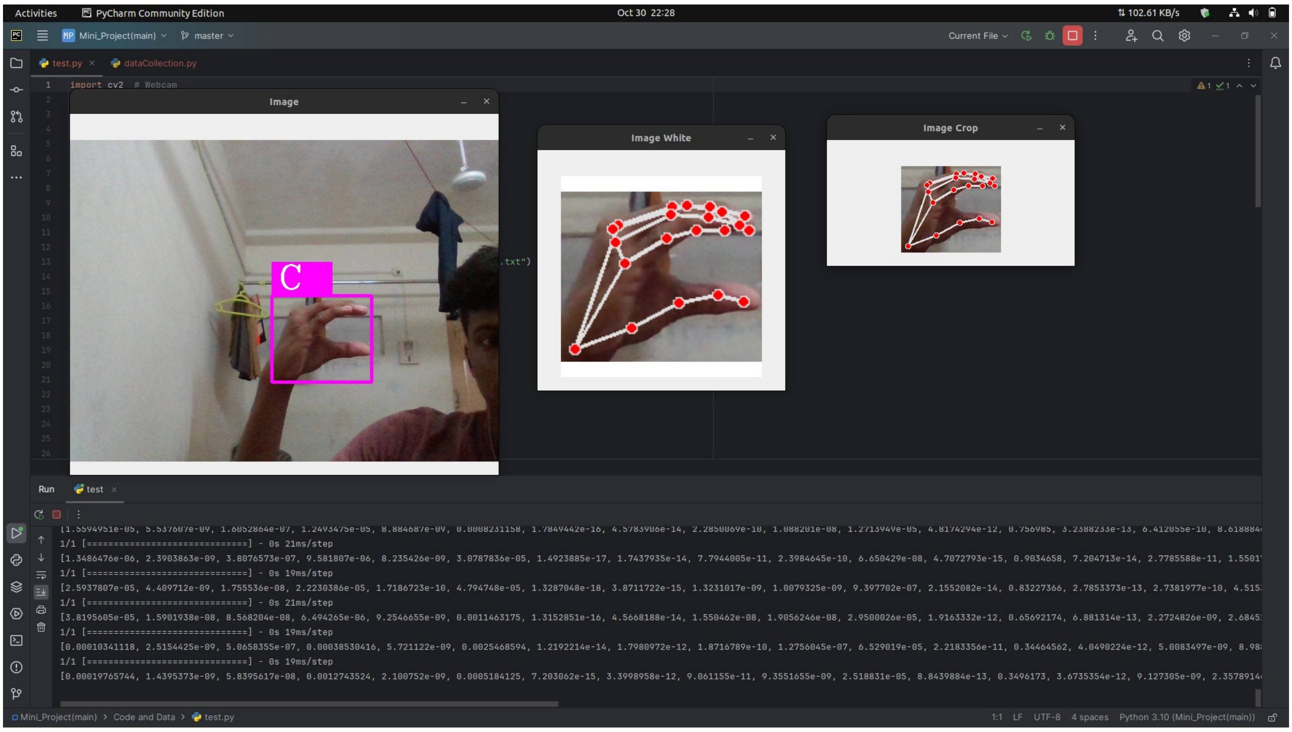


```
import cv2  # Webcam
from cvzone.HandTrackingModule import HandDetector  # Hand Detection
from cvzone.ClassificationModule import Classifier  # Importing Classifier


import numpy as np
import math

# INITIALIZING THE WEBCAM
# Capture object:
cap = cv2.VideoCapture(0)  # 0 is the id number for webcam
detector = HandDetector(maxHands=1)  # number of hands to be detected
classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")

offset = 10  # The value for spacing for exactly cropping the image
imgSize = 300  # The fixed size of the image


labels = ["0", "1", "2", "3", "4", "5",
          "6", "7", "8", "9", "A", "B",
          "C", "D", "E", "F", "G", "H",
          "I", "J", "K", "L", "M", "N",
          "O", "P", "Q", "R", "S", "T",
          "U", "V", "W", "X", "Y", "Z"]
```



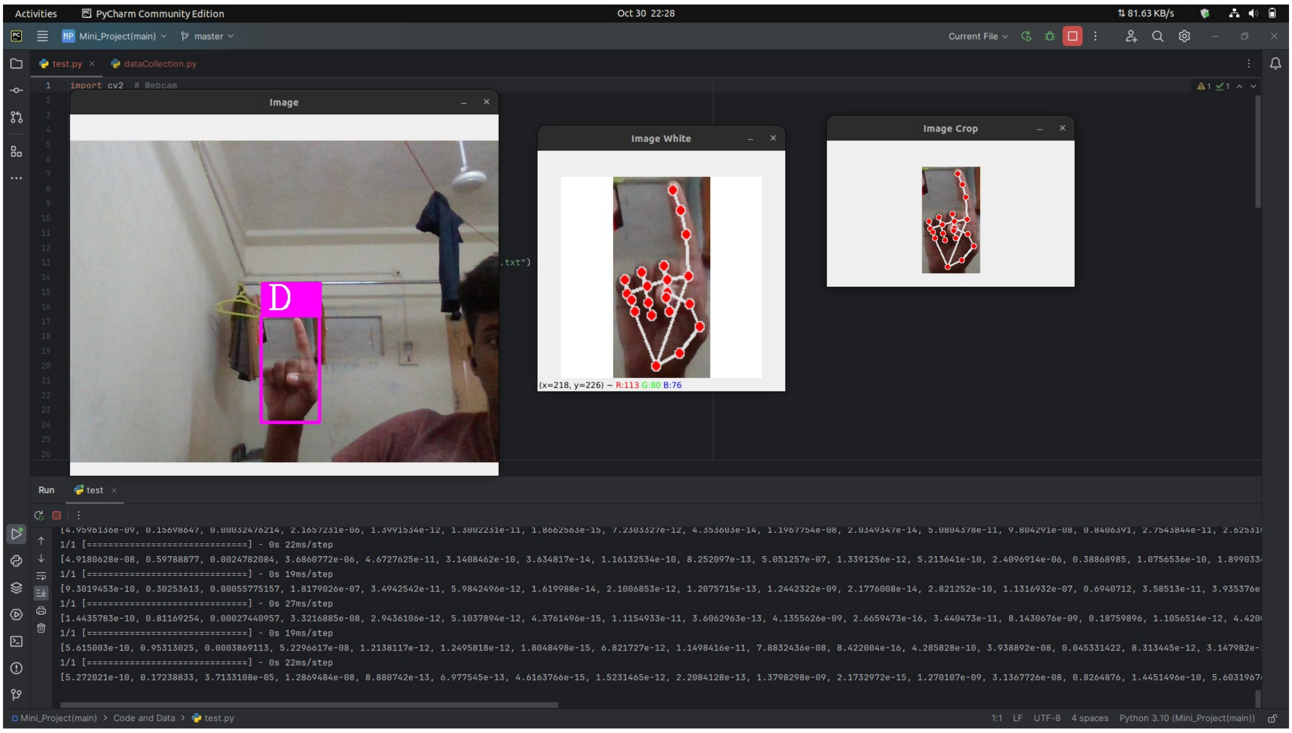
```
while True:
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img = detector.findHands(img)
    if hands:
        hand = hands[0]  # Only one hand
        x, y, w, h = hand['bbox']  # Bounding box: x, y, width and height

        # Creating a background image:
        imgWhite = np.ones((imgSize, imgSize, 3),
                           np.uint8) * 255  # Creating an image of size 300x300␣
 ↪by entering the datatype
        # unsigned integers of 8 bit becoz the image will of 0 to 255

        # Image Crop:
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]

        # Putting the cropped image into the white background:
        imgCropShape = imgCrop.shape

        aspectRatio = h / w
        # Adjusting the width of the image, so it can be in centre
        if aspectRatio > 1:
            k = imgSize / h
            wCal = math.ceil(k * w)
            imgResize = cv2.resize(imgCrop, (wCal, imgSize))
            imgResizeShape = imgResize.shape
            wGap = math.ceil((imgSize - wCal) / 2)
            imgWhite[:, wGap:wCal + wGap] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)

        # Adjusting the height of the image, so it can be in centre
        else:
            k = imgSize / w
            hCal = math.ceil(k * h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCal))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize - hCal) / 2)
            imgWhite[hGap:hCal + hGap, :] = imgResize
            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)

        cv2.rectangle(imgOutput, (x-offset, y-offset-50), (x-offset+90,␣
↪y-offset-50+50), (255, 0, 255), cv2.FILLED)
        cv2.putText(imgOutput, labels[index], (x, y-20), cv2.
↪FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
        cv2.rectangle(imgOutput, (x-offset, y-offset), (x+w+offset, y+h+offset),␣
↪(255, 0, 255), 4)

        cv2.imshow("Image Crop", imgCrop)
        cv2.imshow("Image White", imgWhite)

    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1)
```

Results



```
import cv2  # Webcam
.txt")
```

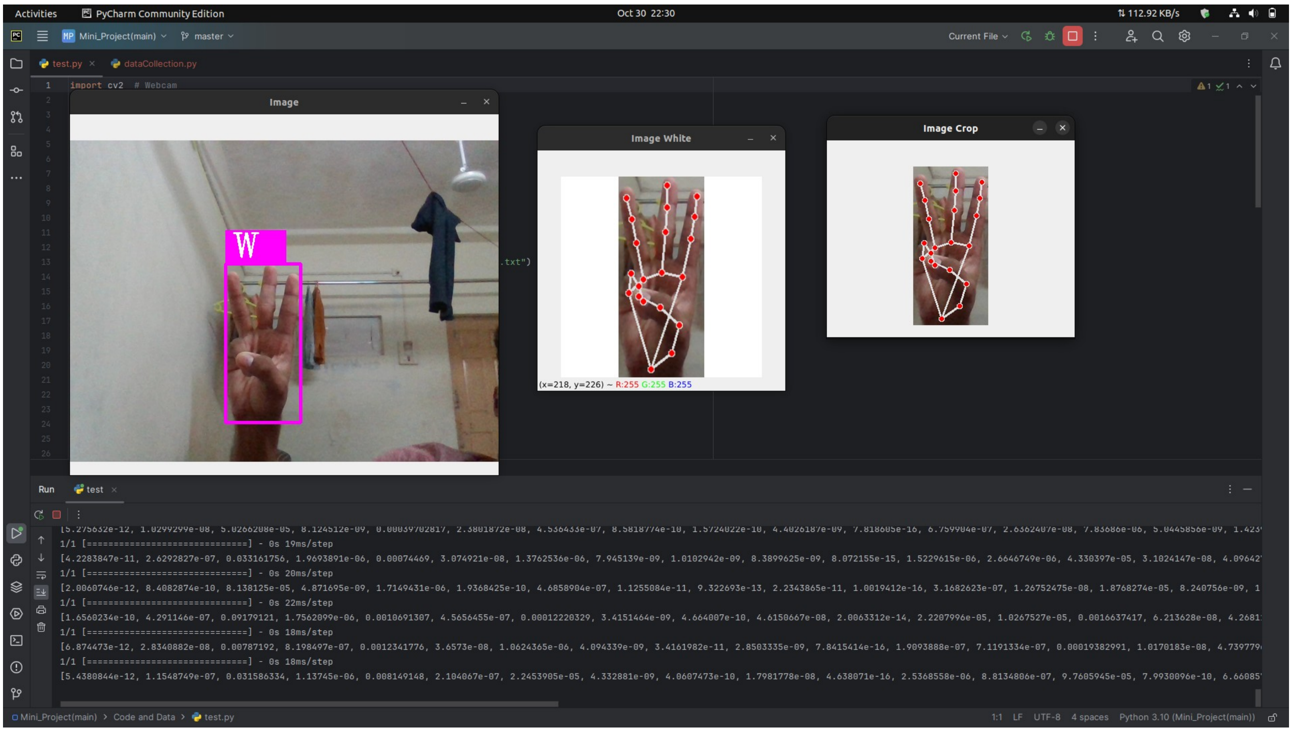
```
[1.5594951e-05, 5.537607e-09, 1.6052864e-07, 1.2493475e-05, 8.884687e-09, 0.0008231158, 1.7849442e-16, 4.5783906e-14, 2.2850069e-10, 1.088201e-08, 1.2713949e-05, 4.8174294e-12, 0.756985, 3.2388233e-13, 6.412055e-10, 8.618884
1/1 [==============================] - 0s 21ms/step
[1.3486476e-06, 2.3903863e-09, 3.8076573e-07, 9.581807e-06, 8.235426e-09, 3.0787836e-05, 1.4923885e-17, 1.7437935e-14, 7.7944005e-11, 2.3984645e-10, 6.650429e-08, 4.7072793e-15, 0.9034658, 7.204713e-14, 2.7785588e-11, 1.5501
1/1 [==============================] - 0s 19ms/step
[2.5937807e-05, 4.409712e-09, 1.755536e-08, 2.2230386e-05, 1.7186723e-10, 4.794748e-05, 1.3287048e-18, 3.8711722e-15, 1.3231017e-09, 1.0079325e-09, 9.397702e-07, 2.1552082e-14, 0.83227366, 2.7853373e-13, 2.7381977e-10, 4.515
1/1 [==============================] - 0s 21ms/step
[3.8195605e-05, 1.5901938e-08, 8.568204e-08, 6.494265e-06, 9.2546655e-09, 0.0011463175, 1.3152851e-16, 4.5668188e-14, 1.550462e-08, 1.9056246e-08, 2.9500026e-05, 1.9163332e-12, 0.65692174, 6.881314e-13, 2.2724826e-09, 2.6845
1/1 [==============================] - 0s 19ms/step
[0.00010341118, 2.5154425e-09, 5.0658355e-07, 0.00038530416, 5.721122e-09, 0.0025468594, 1.2192214e-14, 1.7980972e-12, 1.8716789e-10, 1.2756045e-07, 6.529019e-05, 2.2183356e-11, 0.34464562, 4.0490224e-12, 5.0083497e-09, 8.98
1/1 [==============================] - 0s 19ms/step
[0.00019765744, 1.4395373e-09, 5.8395617e-08, 0.0012743524, 2.100752e-09, 0.0005184125, 7.203062e-15, 3.3998958e-12, 9.061155e-11, 9.3551655e-09, 2.518831e-05, 8.8439884e-13, 0.3496173, 3.6735354e-12, 9.127305e-09, 2.3578914
```



Activities
PyCharm Community Edition
Oct 30 22:28
81.63 KB/s
Mini_Project(main)
master
Current File
test.py
dataCollection.py
Image
Image White
Image Crop
D
(x=218, y=226) ~ R:113 G:80 B:76
Run
test
Mini_Project(main) › Code and Data › test.py
1:1 LF UTF-8 4 spaces Python 3.10 (Mini_Project(main))

```
import cv2  # Webcam
.txt")
```

```
[4.9596136e-09, 0.15698647, 0.00032476214, 2.1657231e-06, 1.3991534e-12, 1.3002231e-11, 1.8662563e-15, 7.2303327e-12, 4.353603e-14, 1.1967754e-08, 2.0349347e-14, 5.0804378e-11, 9.804291e-08, 0.8406391, 2.7543844e-11, 2.625310
1/1 [==============================] - 0s 22ms/step
[4.9180628e-08, 0.59788877, 0.0024782084, 3.6860772e-06, 4.6727625e-11, 3.1408462e-10, 3.634817e-14, 1.16132534e-10, 8.252097e-13, 5.051257e-07, 1.3391256e-12, 5.213641e-10, 2.4096914e-06, 0.38868985, 1.0756536e-10, 1.899033
1/1 [==============================] - 0s 19ms/step
[9.3019453e-10, 0.30253613, 0.00055775157, 1.8179026e-07, 3.4942542e-11, 5.9842496e-12, 1.619988e-14, 2.1006853e-12, 1.2075715e-13, 1.2442322e-09, 2.1776008e-14, 2.821252e-10, 1.1316932e-07, 0.6940712, 3.58513e-11, 3.935376e
1/1 [==============================] - 0s 27ms/step
[1.4435783e-10, 0.81169254, 0.00027440957, 3.3216885e-08, 2.9436106e-12, 5.1037894e-12, 4.3761496e-15, 1.1154933e-11, 3.6062963e-13, 4.1355626e-09, 2.6659473e-16, 3.440473e-11, 8.1430676e-09, 0.18759896, 1.1056514e-12, 4.420
1/1 [==============================] - 0s 19ms/step
[5.615003e-10, 0.95313025, 0.0003869113, 5.2296617e-08, 1.2138117e-12, 1.2495818e-12, 1.8048498e-15, 6.821727e-12, 1.1498416e-11, 7.8832436e-08, 8.422004e-16, 4.285828e-10, 3.938892e-08, 0.045331422, 8.313445e-12, 3.147982e-
1/1 [==============================] - 0s 22ms/step
[5.272021e-10, 0.17238833, 3.7133108e-05, 1.2869484e-08, 8.880742e-13, 6.977545e-13, 4.6163766e-15, 1.5231465e-12, 2.2084128e-13, 1.3798298e-09, 2.1732972e-15, 1.270107e-09, 3.1367726e-08, 0.8264876, 1.4451496e-10, 5.6031967
```


Activities
PyCharm Community Edition
Oct 30 22:30
112.92 KB/s
Mini_Project(main)
master
Current File
test.py
dataCollection.py
Image
W
Image White
(x=218, y=226) ~ R:255 G:255 B:255
Image Crop
Run
test
Mini_Project(main) > Code and Data > test.py
1:1 LF UTF-8 4 spaces Python 3.10 (Mini_Project(main))

```
import cv2  # Webcam
```

```
[5.275632e-12, 1.0299299e-08, 5.0266208e-05, 8.124512e-09, 0.00039702817, 2.3801872e-08, 4.536433e-07, 8.5818774e-10, 1.5724022e-10, 4.4026187e-09, 7.818605e-16, 6.759904e-07, 2.6362407e-08, 7.83686e-06, 5.0445856e-09, 1.423
1/1 [==============================] - 0s 19ms/step
[4.2283847e-11, 2.6292827e-07, 0.033161756, 1.9693891e-06, 0.00074469, 3.074921e-08, 1.3762536e-06, 7.945139e-09, 1.0102942e-09, 8.3899625e-09, 8.072155e-15, 1.5229615e-06, 2.6646749e-06, 4.330397e-05, 3.1024147e-08, 4.09642
1/1 [==============================] - 0s 20ms/step
[2.0060746e-12, 8.4082874e-10, 8.138125e-05, 4.871695e-09, 1.7149431e-06, 1.9368425e-10, 4.6858904e-07, 1.1255084e-11, 9.322693e-13, 2.2343865e-11, 1.0019412e-16, 3.1682623e-07, 1.26752475e-08, 1.8768274e-05, 8.240756e-09, 1
1/1 [==============================] - 0s 22ms/step
[1.6560234e-10, 4.291146e-07, 0.09179121, 1.7562099e-06, 0.0010691307, 4.5656455e-07, 0.00012220329, 3.4151464e-09, 4.664007e-10, 4.6150667e-08, 2.0063312e-14, 2.2207996e-05, 1.0267527e-05, 0.0016637417, 6.213628e-08, 4.2681
1/1 [==============================] - 0s 18ms/step
[6.874473e-12, 2.8340882e-08, 0.00787192, 8.198497e-07, 0.0012341776, 3.6573e-08, 1.0624365e-06, 4.094339e-09, 3.4161982e-11, 2.8503335e-09, 7.8415414e-16, 1.9093888e-07, 7.1191334e-07, 0.00019382991, 1.0170183e-08, 4.739779
1/1 [==============================] - 0s 18ms/step
[5.4380844e-12, 1.1548749e-07, 0.031586334, 1.13745e-06, 0.008149148, 2.104067e-07, 2.2453905e-05, 4.332881e-09, 4.0607473e-10, 1.7981778e-08, 4.638071e-16, 2.5368558e-06, 8.8134806e-07, 9.7605945e-05, 7.9930096e-10, 6.66085
```

Conclusion

In this paper, deep learning convolutional neural network based hand gesture detection and recognition methodology
is proposed. This proposed method segments the finger tips
from the hand gesture image, and then, this finger tips are
given as input to the CNN classifier.

The CNN classification approach trains and classifies the test hand gesture image which is obtained from open access image dataset. The performance of the proposed hand gesturedetection and recognition methodology is analyzed in terms of sensitivity, specificity, accuracy and recognition rate. The proposed hand gesture detection and recognition methodology using CNN classification approach stated in this paper achieves 98.1% of sensitivity, 93.4% of specificity, 96.2% of accuracy and 96.2% of recognition rate.

This Algorithm is Conclusively better than the older one.