**CODE:**

**DataCollection.py:**

import cv2

from cvzone.HandTrackingModule import HandDetector import numpy as np

import math import time

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1) offset = 20

imgSize = 300

folder = "D:/Programming/2) Mango Python/6) Experimental/Project - 4/Numbers/5" counter = 0

while True:

success, img = cap.read() #

hands, img = detector.findHands(img) # if hands:

hand = hands[0]

x, y, w, h = hand['bbox']

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) \* 225 imgCrop = img[y - offset: y + h + offset, x - offset: x + w + offset] imgCropShape = imgCrop.shape

aspectRatio = h / w if aspectRatio > 1: k = imgSize / h

wCal = math.ceil(k \* w)

imgResize = cv2.resize(imgCrop, (wCal, imgSize)) imgResizeShape = imgResize.shape

imgWhite[0:imgResizeShape[0], 0:imgResizeShape[1]] = imgResize wGap = math.ceil((imgSize-wCal)/2)

imgWhite[:, wGap:wCal+wGap] = imgResize

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("ImageCrop", imgCrop)

# cv2.imshow("ImageWhite", imgWhite) cv2.imshow("Image", img)

key = cv2.waitKey(1) if key == ord("s") :

counter += 1 cv2.imwrite(f'{folder}/Image\_{time.time()}.jpg',imgWhite) print(counter)

**Test.py:**

import cv2

from cvzone.HandTrackingModule import HandDetector from cvzone.ClassificationModule import Classifier import numpy as np

import math import time

cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)

classifier = Classifier("D:/Programming/2)Python/6) Experimental/Project - 4/Model - Numbers/converted\_keras/keras\_model.h5","D:/Programming/2)Python/6) Experimental/Project - 4/Model - Numbers/converted\_keras/labels.txt")

#classifier1 = Classifier("D:/Programming/2) Mango Python/6) Experimental/Project - 4/Model - E, F, G, H/converted\_keras/keras\_model.h5","D:/Programming/2) Mango Python/6) Experimental/Project - 4/Model - E, F, G, H/converted\_keras/labels.txt")

offset = 20

imgSize = 300

folder = "D:/Programming/2)Python/6) Experimental/Project - 4/Model - Numbers" counter = 0

# labels = ['Danger','Slow','Stop','Straight','U-Turn'] # labels =

['A',"B","C","D","E","F","G","H","I","J","K","L","O","P","R","S","T","U","V","W","Y"] labels = ["0","1","2","3","4","5"]

# labels = ['stop'] while True:

success, img = cap.read() imgOutput = img.copy()

hands, img = detector.findHands(img) if hands:

hand = hands[0]

x, y, w, h = hand['bbox']

imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) \* 225 imgCrop = img[y - offset: y + h + offset, x - offset: x + w + offset]

imgCropShape = imgCrop.shape aspectRatio = h / w

if aspectRatio > 1: k = imgSize / h

wCal = math.ceil(k \* w)

imgResize = cv2.resize(imgCrop, (wCal, imgSize)) imgResizeShape = imgResize.shape

imgWhite[0:imgResizeShape[0], 0:imgResizeShape[1]] = imgResize wGap = math.ceil((imgSize-wCal)/2)

imgWhite[:, wGap:wCal+wGap] = imgResize

prediction, index = classifier.getPrediction(imgWhite,draw = False) #prediction, index = classifier1.getPrediction(imgWhite, draw=False) print(prediction, index)

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) #prediction, index = classifier1.getPrediction(imgWhite, draw=False)

# 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 - 26), 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("ImageCrop", imgCrop)

# cv2.imshow("ImageWhite", imgWhite) cv2.imshow("Image", imgOutput) cv2.waitKey(1)