Anexă

Codul folosit pentru rezolvarea temei

Data Collection.py – *folosind acest cod am obținut imaginile pentru antrenarea modelului*

```
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 = "C:/Users/alior/Desktop/proiect SVA/Date/D"
contor = 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) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w

        if imgCropShape[0] != 0 and imgCropShape[1] != 0:
            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

            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

        if imgCropShape[0] > 0 and imgCropShape[1] > 0:
            cv2.imshow("ImageCrop", imgCrop)
            cv2.imshow("ImageWhite", imgWhite)

    cv2.imshow("Image", img)
    key = cv2.waitKey(1)
    if key == ord("s"):
        contor += 1
        cv2.imwrite(f'C:/Users/alior/Desktop/proiect
SVA/Date/A/Image_{time.time()}.jpg', imgWhite)
        print(contor)
```

```
    if key == ord("q"):
        break
```

Testing.py *– detectarea mâinii și afișarea mesajului identificat*

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import tkinter as tk
import numpy as np
import math

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier("C:/Users/alior/Desktop/proiect
SVA/Model/keras_model.h5", "C:/Users/alior/Desktop/proiect
SVA/Model/labels.txt")

offset = 20
imgSize = 300
labels = ["A", "B", "C", "D"]

window = tk.Tk()
is_running = False  # Indicator pentru a verifica dacă bucla rulează sau nu


def read_values():
    global is_running
    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) * 255
        imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
        imgCropShape = imgCrop.shape
        aspectRatio = h / w

        if imgCropShape[0] != 0 and imgCropShape[1] != 0:
            if aspectRatio > 1:
                k = imgSize / h
                wCal = math.ceil(k * w)
                imgResize = cv2.resize(imgCrop, (wCal, imgSize))
                wGap = math.ceil((imgSize - wCal) / 2)
                imgWhite[:, wGap:wCal + wGap] = imgResize
                prediction, index = classifier.getPrediction(imgWhite,
draw=False)
                print(prediction, index)

            else:
                k = imgSize / w
                hCal = math.ceil(k * h)
                imgResize = cv2.resize(imgCrop, (imgSize, hCal))
                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), (0, 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), (20, 230, 0), 4)

        if imgCropShape[0] > 0 and imgCropShape[1] > 0:
            cv2.imshow("ImageCrop", imgCrop)
            cv2.imshow("ImageWhite", imgWhite)

    cv2.imshow("Image", imgOutput)

    if is_running:
        window.after(1, read_values)  # Repetă apelul funcției la fiecare 1
milisecundă


def button_pressed():
    global is_running
    if is_running:
        is_running = False
        window.quit()  # Închide fereastra tkinter
    else:
        is_running = True
        read_values()


button = tk.Button(window, text="Apasă", command=button_pressed)
button.pack()

window.mainloop()

cap.release()
cv2.destroyAllWindows()
```