```
#@markdown We implemented some functions to visualize the face landmark
detection results. <br/> Run the following cell to activate the functions.

from mediapipe import solutions
from mediapipe.framework.formats import landmark_pb2
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Mozaikleme fonksiyonu
def apply_mosaic(image, x, y, w, h, mosaic_scale=10):
    # Koordinatları görüntü sınırları içinde tut
    x = max(0, int(x))
    y = max(0, int(y))
    w = min(int(w), image.shape[1] - x)
    h = min(int(h), image.shape[0] - y)

    if w <= 0 or h <= 0:
        print(f"Geçersiz mozaik bölgesi: x={x}, y={y}, w={w}, h={h}")
        return image  # Geçersiz bölge, mozaik uygulanmaz

    # Mozaik yapılacak bölgeyi al
    roi = image[y:y+h, x:x+w]
    # Bölgeyi küçült
    small_w, small_h = max(1, w // mosaic_scale), max(1, h // mosaic_scale)  #
Sıfır bölmeyi önle
    small = cv2.resize(roi, (small_w, small_h),
interpolation=cv2.INTER_LINEAR)
    # Tekrar büyüt (tam orijinal boyutlara)
    mosaic = cv2.resize(small, (w, h), interpolation=cv2.INTER_NEAREST)
    # Mozaiklenmiş bölgeyi orijinal görüntüye yerleştir
    image[y:y+h, x:x+w] = mosaic
    return image

def draw_landmarks_on_image(rgb_image, detection_result):
    # Bulunan yüzler ve o yüzler üzerindeki koordinatlar
    face_landmarks_list = detection_result.face_landmarks
    annotated_image = np.copy(rgb_image)

    # Yüz sayısı
    for idx in range(len(face_landmarks_list)):
        face_landmarks = face_landmarks_list[idx]

        # Yüzün sınırlarını (bounding box) hesapla
        x_coords = [landmark.x * rgb_image.shape[1] for landmark in
face_landmarks]
```

```
        y_coords = [landmark.y * rgb_image.shape[0] for landmark in
face_landmarks]
        x_min, x_max = min(x_coords), max(x_coords)
        y_min, y_max = min(y_coords), max(y_coords)
        w, h = x_max - x_min, y_max - y_min

        # Yüz bölgesine mozaik uygula
        annotated_image = apply_mosaic(annotated_image, x_min, y_min, w, h,
mosaic_scale=15)

        # Koordinatları CSV'ye kaydetme
        koordinatlar = []
        for landmark in face_landmarks:
            koordinatlar.append(str(round(landmark.x, 4)))
            koordinatlar.append(str(round(landmark.y, 4)))

        koordinatlar = ",".join(koordinatlar)
        koordinatlar += f",{etiket}\n"
        with open("veriseti.csv", "a") as f:
            f.write(koordinatlar)

    return annotated_image

def plot_face_blendshapes_bar_graph(face_blendshapes):
    face_blendshapes_names = [face_blendshapes_category.category_name for
face_blendshapes_category in face_blendshapes]
    face_blendshapes_scores = [face_blendshapes_category.score for
face_blendshapes_category in face_blendshapes]
    face_blendshapes_ranks = range(len(face_blendshapes_names))

    fig, ax = plt.subplots(figsize=(12, 12))
    bar = ax.barh(face_blendshapes_ranks, face_blendshapes_scores,
label=[str(x) for x in face_blendshapes_ranks])
    ax.set_yticks(face_blendshapes_ranks, face_blendshapes_names)
    ax.invert_yaxis()

    for score, patch in zip(face_blendshapes_scores, bar.patches):
        plt.text(patch.get_x() + patch.get_width(), patch.get_y(),
f"{score:.4f}", va="top")

    ax.set_xlabel('Score')
    ax.set_title("Face Blendshapes")
    plt.tight_layout()
    plt.show()

def sutun_basliklarini_olustur():
```

```
    with open("veriseti.csv", "w") as f:
        satir = ""
        for i in range(1, 479):
            satir = satir + f"x{i},y{i},"
        satir = satir + "Etiket\n"
        f.write(satir)

etiket = "happy"
# NOT: aşağıdaki fonksiyon sadece ilk etiket için çalışacak
# diğer etiketlerde bu satırı yorum satırı haline getirin
sutun_basliklarini_olustur()

# STEP 1: Gerekli modülleri import et
import mediapipe as mp
from mediapipe.tasks import python
from mediapipe.tasks.python import vision

# STEP 2: FaceLandmarker nesnesi oluştur
base_options =
python.BaseOptions(model_asset_path='face_landmarker_v2_with_blendshapes.task'
)
options = vision.FaceLandmarkerOptions(
    base_options=base_options,
    output_face_blendshapes=True,
    output_facial_transformation_matrixes=True,
    num_faces=5  # Maksimum 5 yüzü algıla
)
detector = vision.FaceLandmarker.create_from_options(options)

# Kameradan görüntü alımı
cam = cv2.VideoCapture(0)  # Parantez düzeltildi
while cam.isOpened():
    basari, frame = cam.read()
    if not basari:
        print("Kamera görüntüsü alınamadı!")
        break

    try:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=frame)

        # Yüz landmarklarını tespit et
        detection_result = detector.detect(mp_image)

        # Algılama sonuçlarını işle ve görselleştir
```

```
        annotated_image = draw_landmarks_on_image(mp_image.numpy_view(),
detection_result)
        cv2.imshow("yuz", cv2.cvtColor(annotated_image, cv2.COLOR_RGB2BGR))

        key = cv2.waitKey(1)
        if key == ord('q') or key == ord('Q'):
            break
    except Exception as e:
        print(f"Hata oluştu: {e}")
        continue

# Kaynakları serbest bırak
cam.release()
cv2.destroyAllWindows()
```