Міністерство освіти і науки України Національний технічний університет України «КПІ» імені Ігоря Сікорського Кафедра інформатики та програмної інженерії ФІОТ

ЗВІТ з лабораторної роботи №5 з навчальної дисципліни «Технології Computer Vision»

Тема: РЕАЛІЗАЦІЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ (MACHINE LEARNING)

Виконав
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-12
Васильєв Є.К.

Перевірив
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2023

І. Мета: Виявити дослідити та узагальнити особливості аналізу даних з використанням методів та технологій машинного навчання (Machine Learning (ML)).

ІІ. Завдання:

Група технічних вимог_3:

Підрахувати кількість об’єктів на обраному цифровому зображенні. Об’єкти, що підлягають обрахунку обрати самостійно. Зміст етапів попередньої обробки зображень (корекція кольору, фільтрація, векторизація, кластеризація) має буди результатом R&D процесів, що конкретизується обраним зображенням і об’єктами для підрахунку. Провести аналіз отриманих результатів, сформувати висновки.

Група технічних вимог_5:

Ідентифікувати об’єкти в обраному відеопотоці. Об’єкти, що підлягають ідентифікації обрати самостійно. Зміст етапів попередньої обробки відеопотоку (корекція кольору, фільтрація, векторизація, кластеризація (за необхідності)) має буди результатом R&D процесів, що конкретизується обраним відео і об’єктом ідентифікації. Провести аналіз отриманих результатів, сформувати висновки.

ІІІ. Результати виконання лабораторної роботи.

3.1. Синтезована математична модель;

Для виявлення об'єктів на фото було обрано MediaPipe - відкриту бібліотека та набір інструментів для роботи з комп'ютерним зором та обробкою зображень. Розроблена Google, MediaPipe надає простий спосіб виявлення та відстеження різних об'єктів на відео та відеопотоках. Mediapipe складається з набору готових модулів, які можна використовувати для вирішення різних завдань, основними з яких є:

- виявлення об'єктів (Object Detection) - відстеження та позначення об'єктів на зображеннях і відео;
- класифікація зображень (Image Classification) - визначення вмісту зображень і відео;
- розмітка руки (Hand Landmark) – ідентифікація та відстеження рук й пальців;
- сегментація зображень (Image Segmentation) - знаходження об'єктів та створення масок зображень за допомогою міток;
- розпізнавання обличь (Face Detection) - розпізнавання обличь на зображеннях і відео.

3.2. Результати архітектурного проектування та їх опис;

Для детекції об’єктів на фото та відео було обрано дві моделі: EfficientDet-Lite2 та SSD MobileNetV2. Обидві моделі були навчені на COCO (Common Object in Context), великомасштабному наборі даних для виявлення об'єктів, який містить 1,5 мільйона екземплярів об'єктів і 80 міток об'єктів. Різниця у моделях полягає в їх архітектурі, з якої випливають основні характеристики: EfficientDet-Lite2 зазвичай має вищу точність порівняно з SSD MobileNetV2 завдяки більшій кількості параметрів та більшому розміру моделі, однак SSD MobileNetV2 є менш вимогливою до ресурсів та має кращу швидкість роботи.

3.3. Опис структури проекту програми;

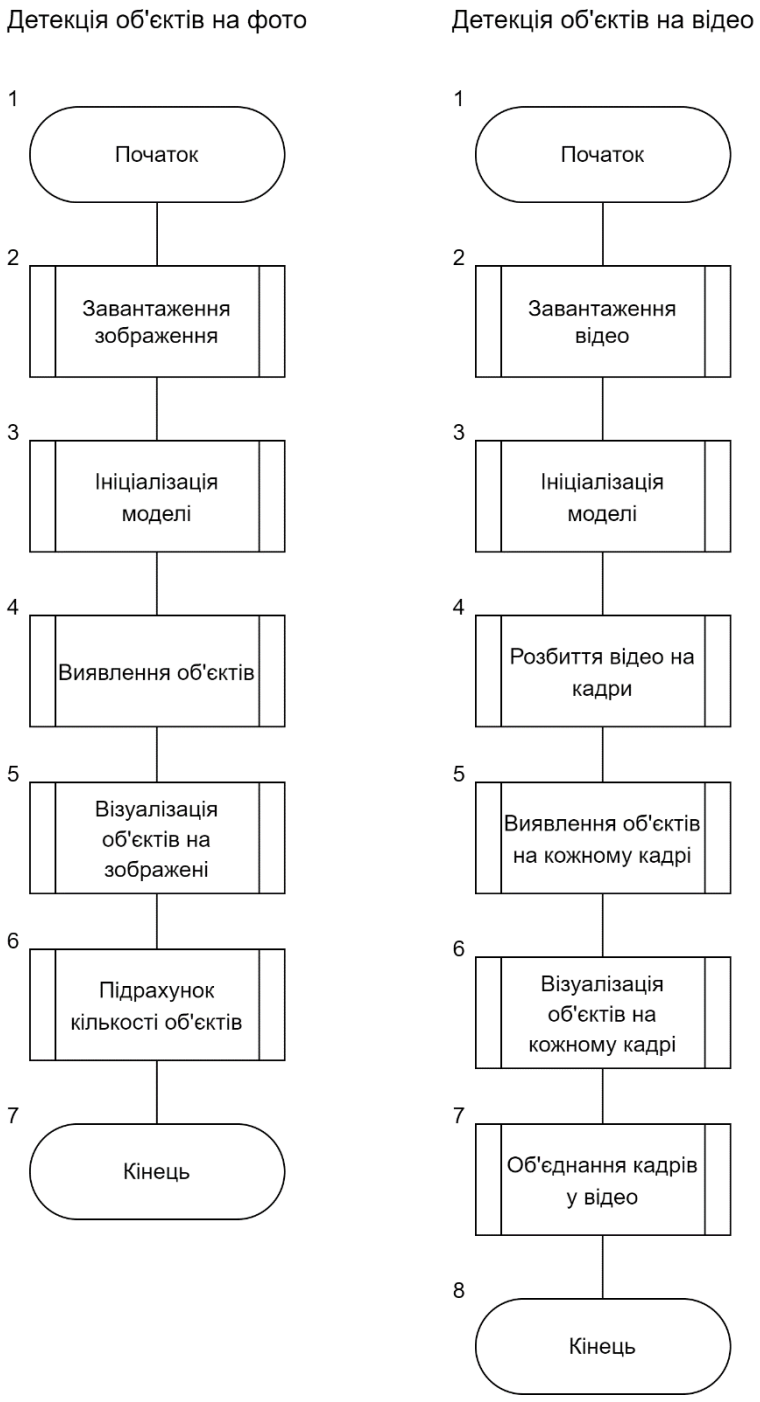


Рис.1. Блок схема алгоритму програми

3.4. Результати роботи програми відповідно до завдання (допускається у формі скриншотів);

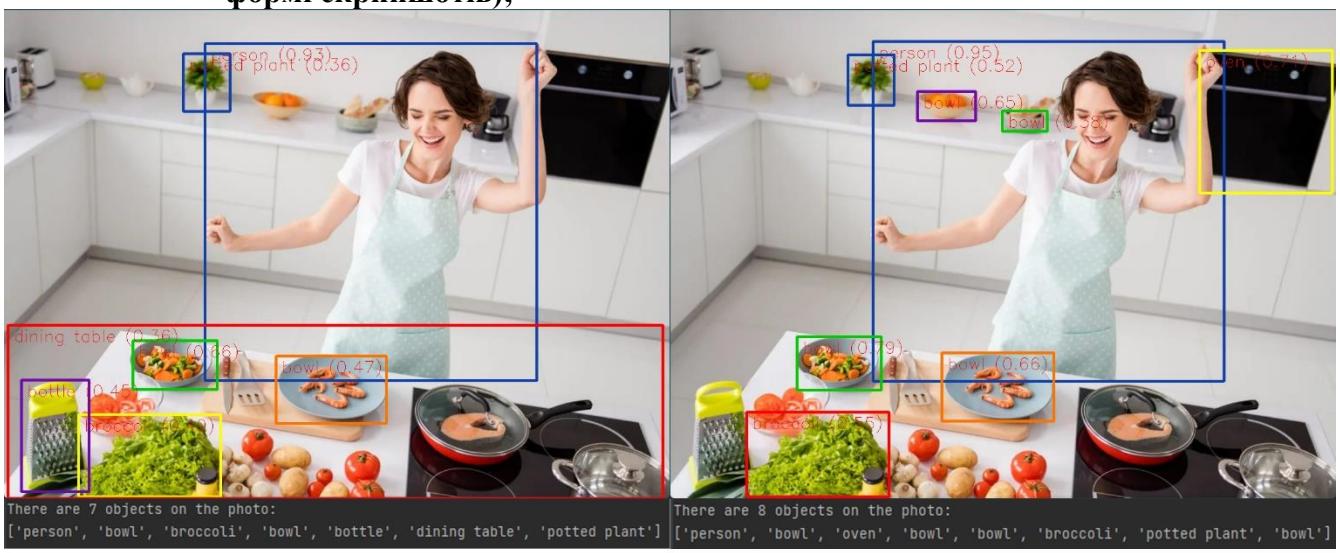


Рис.2. Порівняння моделей детекції об’єктів (фото)

Рис.3. Порівняння моделей детекції об’єктів (фото)

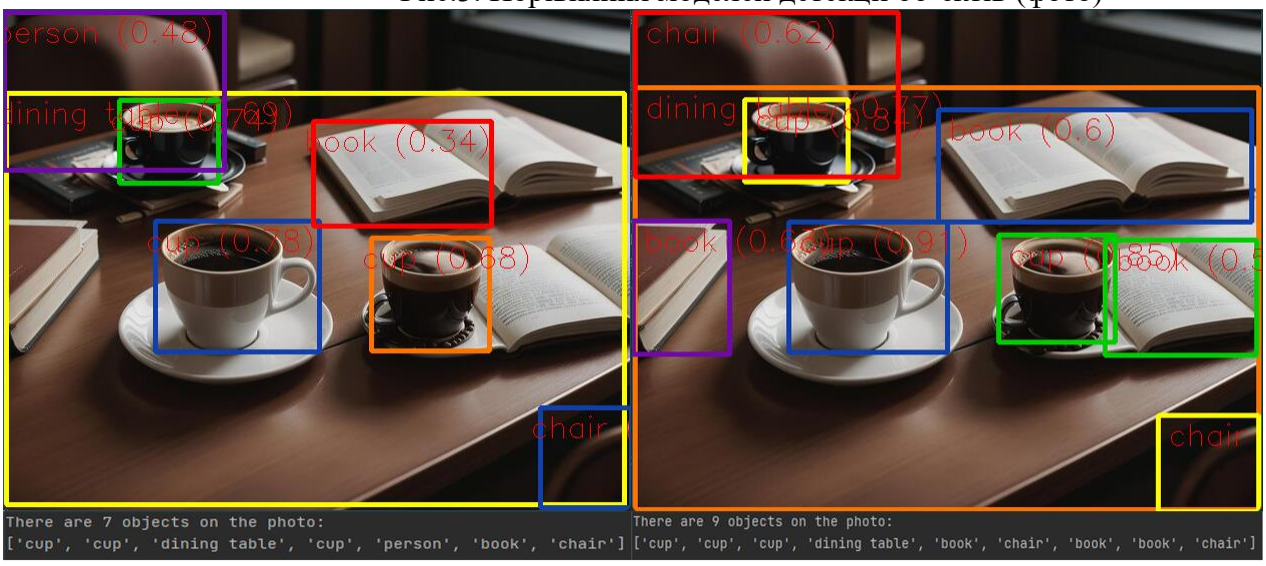
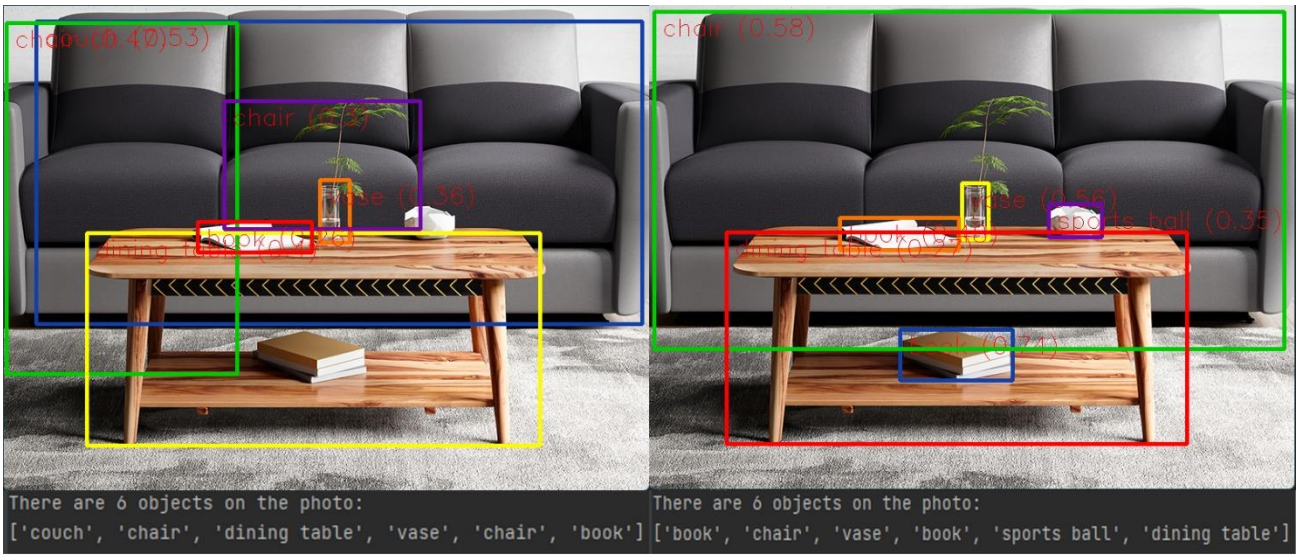
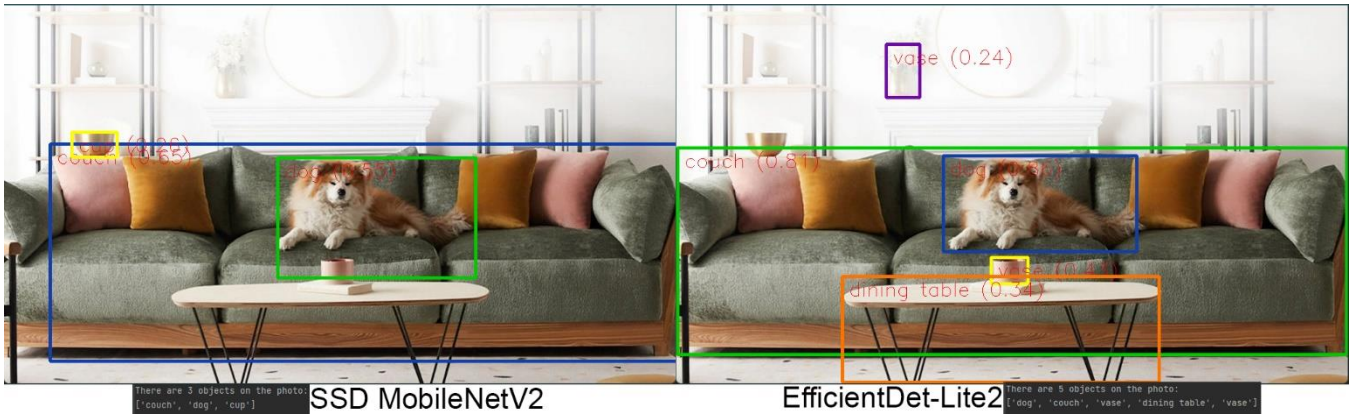


Рис.4. Порівняння моделей детекції об’єктів (фото)



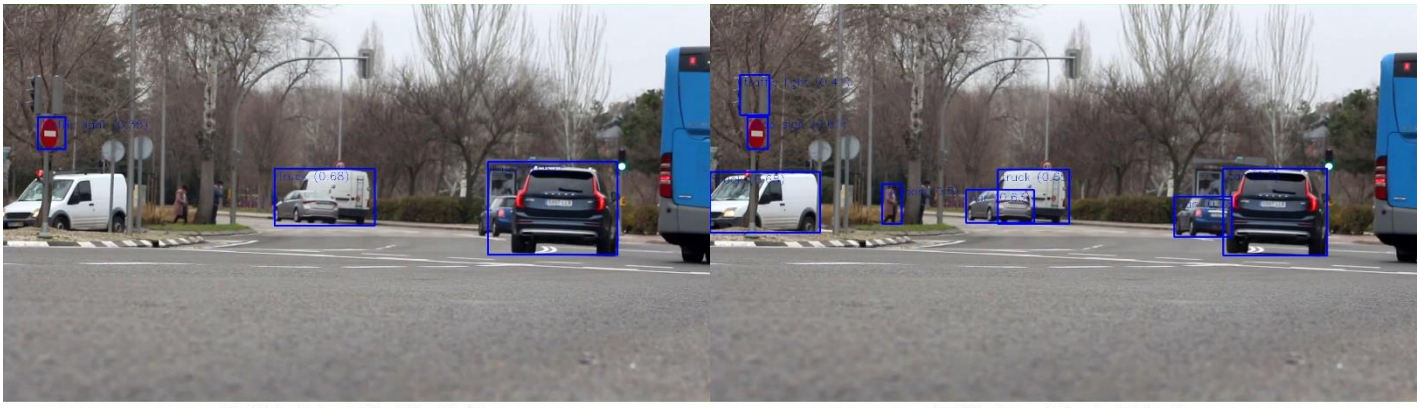
SSD MobileNetV2 EfficientDet-Lite2

Рис.5. Порівняння моделей детекції об’єктів (фото)



SSD MobileNetV2 EfficientDet-Lite2

Рис.6. Порівняння моделей детекції об’єктів (фото)



SSD MobileNetV2 EfficientDet-Lite2

Рис.7. Порівняння моделей детекції об’єктів (довільний скріншот із відео)

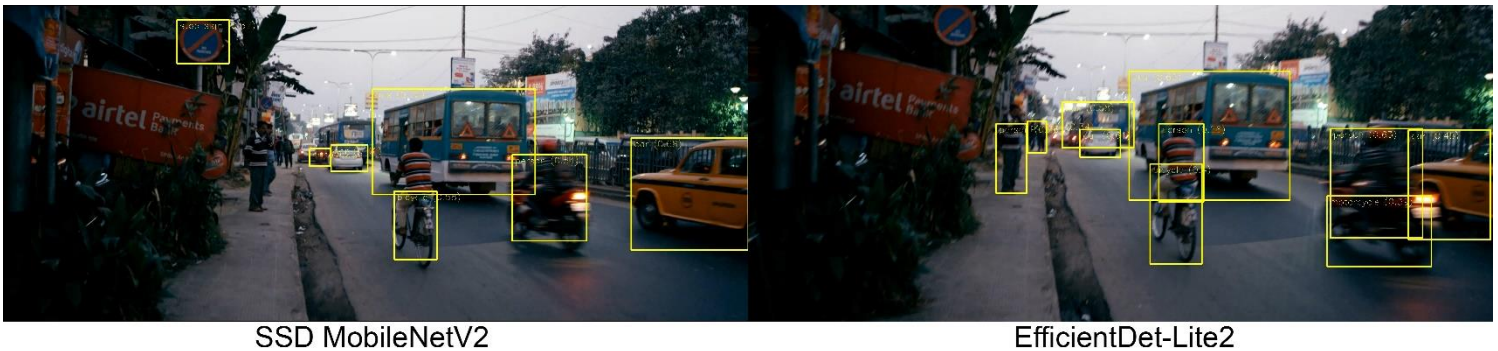


Рис.8. Порівняння моделей детекції об’єктів (довільний скріншот із відео)

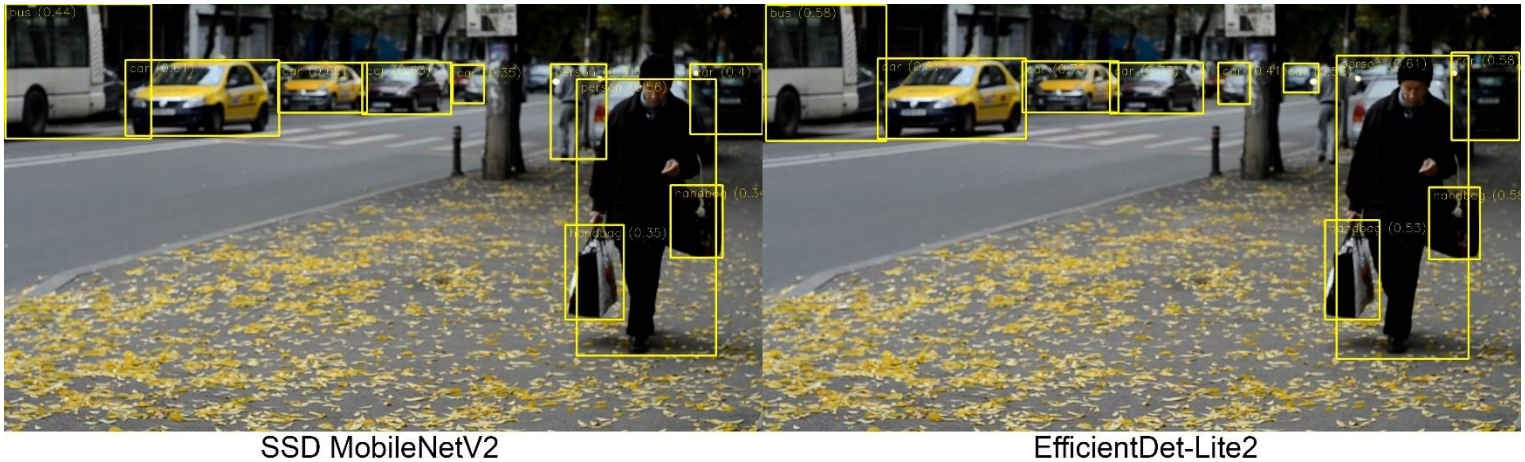


Рис.9. Порівняння моделей детекції об’єктів (довільний скріншот із відео)

3.5. Програмний код, що забезпечує отримання результату (допускається у формі скриншотів).

```python
import ...

MARGIN = 10  # pixels
ROW_SIZE = 20  # pixels
FONT_SIZE = 1
FONT_THICKNESS = 1
RECTANGLES_COLORS = [(17, 64, 170), (0, 203, 0), (253, 252, 1), (252, 115, 0), (113, 8, 170), (254, 0, 0)]
TEXT_COLOR = (0, 255, 255)

model_path = r'C:/Users/egorv/PycharmProjects/pythonProject/Data science/Lab5/efficientdet_lite2.tflite'
IMAGE_FILE = r'C:/Users/egorv/PycharmProjects/pythonProject/Data science/Lab5/dog-and-sofa.jpg'
VIDEO_FILE = r'C:/Users/egorv/PycharmProjects/pythonProject/Data science/Lab5/street2.mp4'


def visualize(image, detection_result):
    """
    Draws bounding boxes on the input image and return it

    Parameters
    ----------
    image: The input RGB image
    detection_result: The list of all "Detection" entities to be visualized

    Returns
    -------
    image: image with bounding boxes
    objects: names of objects on the photo
    """
    objects = []
    for i, detection in enumerate(detection_result.detections):
        # Draw bounding_box
        bbox = detection.bounding_box
        start_point = bbox.origin_x, bbox.origin_y
        end_point = bbox.origin_x + bbox.width, bbox.origin_y + bbox.height
        cv2.rectangle(image, start_point, end_point, TEXT_COLOR, # RECTANGLES_COLORS[i % len(RECTANGLES_COLORS)]
                      , 3)

        # Draw label and score
        category = detection.categories[0]
        category_name = category.category_name
        objects.append(category_name)
        probability = round(category.score, 2)
        result_text = category_name + ' (' + str(probability) + ')'
        text_location = (bbox.origin_x + MARGIN,  # + bbox.width//2
                         MARGIN + ROW_SIZE + bbox.origin_y)
        cv2.putText(image, result_text, text_location, cv2.FONT_HERSHEY_SIMPLEX,
                    FONT_SIZE, TEXT_COLOR, FONT_THICKNESS)

    return image, objects
```

```python
def obj_detection_img(path):
    """
    Detects objects on the photo

    Parameters
    ----------
    path: path to the image file
    """

    # STEP 2: Create an ObjectDetector object.
    base_options = python.BaseOptions(model_asset_path=model_path)
    options = vision.ObjectDetectorOptions(base_options=base_options, score_threshold=0.15, max_results=12)
    detector = vision.ObjectDetector.create_from_options(options)

    # STEP 3: Load the input image.
    image = mp.Image.create_from_file(path)

    # STEP 4: Detect objects in the input image.
    detection_result = detector.detect(image)

    # STEP 5: Process the detection result. In this case, visualize it.
    image_copy = np.copy(image.numpy_view())
    annotated_image, all_objects = visualize(image_copy, detection_result)
    rgb_annotated_image = cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB)

    print(f"There are {len(detection_result.detections)} objects on the photo:")
    print(all_objects)

    cv2.imshow( winname: "Object Detection", rgb_annotated_image)
    cv2.waitKey()


1 usage
def obj_detection_video(path):
    """
    Detects objects on the video

    Parameters
    ----------
    path: path to the video file
    """
    # Load the video file
    VisionRunningMode = mp.tasks.vision.RunningMode
    base_options = python.BaseOptions(model_asset_path=model_path)
    options = vision.ObjectDetectorOptions(base_options=base_options, score_threshold=0.2, max_results=10,
                                           running_mode=VisionRunningMode.VIDEO)
    detector = vision.ObjectDetector.create_from_options(options)

    cap = cv2.VideoCapture(path)
    fps = cap.get(cv2.CAP_PROP_FPS)

    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    size = (frame_width, frame_height)
    result = cv2.VideoWriter('result.avi', cv2.VideoWriter_fourcc(*'MJPG'), fps, size)
```

```
    # Process each frame of the video
    start = time.time()
    while True:
        # Read the next frame from the video
        ret, frame = cap.read()
        if not ret:
            break
        # Press Q on keyboard to  exit
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

        mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=frame)
        frame_index = cap.get(cv2.CAP_PROP_POS_FRAMES)
        # Detect objects in the frame
        frame_timestamp_ms = int(1000 * frame_index / fps)
        detection_result = detector.detect_for_video(mp_image, frame_timestamp_ms)
        annotated_image, _ = visualize(frame, detection_result)
        result.write(frame)
        cv2.imshow( winname: "Object Detection", frame)
    # Close the video capture and all windows
    end = time.time()
    print(f"Total taken {end - start} seconds")
    cap.release()
    result.release()
    cv2.destroyAllWindows()


print('Object tracking: video or image?')
print('1 - Video')
print('2 - Image')
mode = int(input('Mode:'))

if mode == 1:
    print('1 - Video (press q to close the window)')
    obj_detection_video(VIDEO_FILE)

if mode == 2:
    print('2 - Image')
    obj_detection_img(IMAGE_FILE)
```

IV. Висновки.

В ході лабораторної роботи було розроблено програмний скрипт, що реалізує виявлення об'єктів на фото та відео, застосовуючи кроссплатформний фреймворк Mediapipe від Google.

Обидві моделі, що було протестовано, показують гарні результати, та можуть бути використані як основа для тренування власної моделі, адаптованої під конкретні потреби. Для виконання завдання було використано базові моделі та налаштовано їх конфігурації (score_threshold, max_results, category_allow_list, category_deny_list) для отримання оптимального (більшість об’єктів класифіковано правильно) результату.

Загалом, модель EfficientDet-Lite2 проявила кращу точність (іноді навіть ідеальну) порівняно з SSD MobileNetV2, що стосується як кількості виявлених об'єктів, так і правильності їх класифікації, а також впевненості моделі (особливо в ситуаціях, коли кількість об'єктів приблизно однакова та їх мітки збігаються, як це відзначено на скріншоті з останнього відео).

Важливо зазначити, що під час виявлення об'єктів на фотографіях різниця в швидкості роботи моделей майже не помітна, але на відео вона вже значно впливає на продуктивність. У середньому EfficientDet-Lite2 працювала на 140% повільніше, ніж SSD MobileNetV2, і для виділення об'єктів на двадцяти секундному відео знадобилося більше двох хвилин, що може бути критичним в деяких сценаріях.

Номер відео	Тривалість відео, с	Час SSD MobileNetV2, с	Час EfficientDet-Lite2, с	Різниця, %
1	22	50	130	160
2	4	28	62	121
3	39	87	210	141

Усі відео з виділеними об’єктами було завантажено сюди (YouTube).

Виконав: Васильєв Єгор