Додаток А

```
import cv2
import numpy as np
import imutils

protopath = "MobileNetSSD_deploy.prototxt.txt"
modelpath = "MobileNetSSD_deploy.caffemodel"
detector = cv2.dnn.readNetFromCaffe(prototxt=protopath, caffeModel=modelpath)

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
          "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
          "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
          "sofa", "train", "tvmonitor"]

def func1(input):
    image = cv2.imread(input)
    image = imutils.resize(image, width=600)

    (H, W) = image.shape[:2]

    blob = cv2.dnn.blobFromImage(image, 0.007843, (W, H), 127.5)

    detector.setInput(blob)
    person_detections = detector.forward()

    for i in np.arange(0, person_detections.shape[2]):
        confidence = person_detections[0, 0, i, 2]
        if confidence > 0.5:
            idx = int(person_detections[0, 0, i, 1])
```

```
            if CLASSES[idx] != "person":
                continue

            person_box = person_detections[0, 0, i, 3:7] * np.array([W, H, W, H])
            (startX, startY, endX, endY) = person_box.astype("int")

            cv2.rectangle(image, (startX, startY), (endX, endY), (0, 0, 255), 2)

            label = "{}: {:.2f}%".format(CLASSES[idx],
                                         confidence * 100)
            cv2.rectangle(image, (startX, startY), (endX, endY),
                          [idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(image, label, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, [idx], 2)

    cv2.imshow("Results", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def main():
    func1('people_images/people1.jpg')
    func1('people_images/people2.jpg')
    func1('people_images/people3.jpg')
    func1('people_images/people4.jpg')
    func1('people_images/people5.jpg')


main()
```

Додаток Б

```
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
	help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
	help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
	"bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
	"dog", "horse", "motorbike", "person", "pottedplant", "sheep",
	"sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
fps = FPS().start()

while True:
```

```
frame = vs.read()
frame = imutils.resize(frame, width=720)

(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
                        0.007843, (300, 300), 127.5)

net.setInput(blob)
detections = net.forward()

for i in np.arange(0, detections.shape[2]):

    confidence = detections[0, 0, i, 2]

    if confidence > args["confidence"]:
        idx = int(detections[0, 0, i, 1])
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        label = "{}: {:.2f}%".format(CLASSES[idx],
                                confidence * 100)
        cv2.rectangle(frame, (startX, startY), (endX, endY),
                    COLORS[idx], 2)
        y = startY - 15 if startY - 15 > 15 else startY + 15
        cv2.putText(frame, label, (startX, y),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
```

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

    fps.update()

fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

cv2.destroyAllWindows()
vs.stop()
```

Додаток В

```
import cv2
import numpy as np
import pandas as pd
import imutils
import os
import matplotlib.pyplot as plt

# Деплоємо нашу модель та параметри
protopath = "MobileNetSSD_deploy.prototxt.txt"
modelpath = "MobileNetSSD_deploy.caffemodel"
detector = cv2.dnn.readNetFromCaffe(prototxt=protopath, caffeModel=modelpath)

# Задання класів для обробки зображень
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
          "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
          "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
          "sofa", "train", "tvmonitor"]

# Завантаження датасету зображень
dataset_path = 'crowd_dataset/frames/frames'
image_paths = [os.path.join(dataset_path, f) for f in os.listdir(dataset_path) if
               os.path.isfile(os.path.join(dataset_path, f))]

# Створення пустого DataFrame для результатів
results_df = pd.DataFrame(columns=['Image', 'Predicted', 'Count'])



# Функція що перетворює числовий ідентифікатор зображення
# у відносний шлях до файлу, заповнюючи початкові нулі
# та додаючи розширення і каталог файлу.
```

```
def replace_zeros(image_id: int) -> str:
    image_id = str(image_id).rjust(6, '0')
    return f'crowd_dataset/frames/frames/seq_{image_id}.jpg'


# Завантаження labels.csv файлу
labels_file = 'crowd_dataset/labels.csv'

# Вивід даних з датасету
data = pd.read_csv(labels_file)
data['path'] = data['id'].apply(replace_zeros)
print("General output information of the dataset in the amount of 5 initial and 5 final images:")
print(data.head(5))
print(data.tail(5))

# Вивід статистики даних датасету
stats = data.describe()
print("\nOutput of computed dataset statistics:")
print(stats)

# Виводимо статистику датасету за допомогою бібліотеки Matplotlib
plt.hist(data['count'], bins=20)
plt.axvline(stats.loc['mean', 'count'], label='Mean value', color='yellow')
plt.legend()
plt.xlabel('Number of people')
plt.ylabel('Frequency of the number of people')
plt.title('Histogram of the number of silhouette detections')
plt.show()

print("\nClassification of objects has been started, please wait for results...")
```

```
for image_path in image_paths:
    image = cv2.imread(image_path)
    image_size = imutils.resize(image, width=640)

    (H, W) = image_size.shape[:2]

    blob = cv2.dnn.blobFromImage(image_size, 0.007843, (W, H), 127.5)

    detector.setInput(blob)
    person_detections = detector.forward()

    count = 0

    for i in np.arange(0, person_detections.shape[2]):
        confidence = person_detections[0, 0, i, 2]
        if confidence > 0.5:
            idx = int(person_detections[0, 0, i, 1])

            if CLASSES[idx] == "person":
                count = count + 1

    # Отримати ground truth кількість людей з датасету
    image_filename = os.path.basename(image_path)
    image_id = int(os.path.splitext(image_filename)[0].split('_')[1])
    truth_count = data[data['id'] == image_id]['count'].values[0]

    # Додати результати до DataFrame
    results_df = pd.concat([results_df, pd.DataFrame(
        {'Image': [image_path], 'Predicted': [count], 'Count': [truth_count]})],
                           ignore_index=True)
```

```
results_df['MAE'] = (results_df['Count'] - results_df['Predicted']).abs()
results_df['MSE'] = results_df['MAE'] ** 2

plt.hist(results_df['MAE'], bins=20)
plt.xlabel('Absolute Errors')
plt.ylabel('Errors frequency')
plt.title('Histogram of Absolute Errors')
plt.show()

plt.scatter(results_df['Count'], results_df['Predicted'])
plt.xlabel('Actual person count')
plt.ylabel('Predicted person count')
plt.title('Predicted vs Actual Count')
plt.show()

# Збереження результатів обробки зображень у файл .csv
results_df.to_csv('results.csv', index=False)

# Вивести результати
print("\nClassification has been completed:")
print(results_df)

# Обчислити метрики порівняння
diff = results_df['Predicted'] - results_df['Count']
mae = np.abs(diff).mean()
mse = (diff ** 2).mean()
rmse = np.sqrt(mse)

print("\nComparison of the results:")
print(f"Mean Absolute Error (MAE): {mae}")
```

```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Square Error (RMSE): {rmse}")
```

Інструкція щодо запуску програм для rto_and_obj_detection, які були написані для дипломної роботи:

Використаний датасет: https://www.kaggle.com/datasets/fmena14/crowd-counting

Використана версія Python 3.11

1. **Програма obj_detection_image:**

Для роботи програми потрібно встановити такі бібліотеки: cv2, numpy, imutils. Задля їх встановлення потрібно виконати такі команди:

pip install opencv-python;

pip install numpy;

pip install imutils;

Так як в даній програмі вже довільно вставлені зображення для обробки, задля її запуску достатньо буде просто встановити усі бібліотеки та запустити програму за допомогою кнопки запуску в інтегрованому середовищі розробки або консольною командою**:** python obj_detection_image.py

2. **Програма real_time_object_detection:**

Для роботи програми потрібно встановити такі бібліотеки: cv2, numpy, imutils, argparse. Задля їх встановлення потрібно виконати такі команди:

pip install numpy;

pip install argparse;

pip install imutils;

pip install opencv-python;

Запуск програми проводиться лише за вказаним алгоритмом запуску (та слід почекати певний проміжок часу для прогрівання об’єктиву камери, після чого виводиться зображення на екран) встановлюючи завчасно усі бібліотеки та вписуючи в консоль такі дані: python real_time_object_detection.py --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel

3. Програма crowd_object_detection:

Для роботи програми потрібно встановити такі бібліотеки: cv2, numpy, imutils, pandas, os, matplotlib. Задля їх встановлення потрібно виконати такі команди:

pip install opencv-python;

pip install numpy;

pip install imutils;

pip install pandas;

pip install matplotlib

Перед запуском програми слід встановити усі бібліотеки та датасет який вказаний вище та після чого запустити за допомогою кнопки запуску в інтегрованому середовищі розробки або консольною командою**:** python crowd_object_detection.py