Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «Обработка изображений в интеллектуальных системах»

Тема: «Обучение детектеров обьектов»

Выполнил:

Студент 4 курса

Группы ИИ-23

Бусень А.Д.

Проверила:

Андренко К.В.

Брест 2025

Цель работы: осуществить обучение нейросетевого детектора для решения задачи обнарудения заданных объектов.

Общее задание

1. Базируясь на своем варианте, ознакомится с выборкой для обучения детектора, выполнить необходимые преобразования данных для организации процесса обучения (если это нужно!);
2. Для заданной архитектуры нейросетевого детектора организовать процесс обучения для своей выборки. Оценить эффективность обучения на тестовой выборке (mAP);
3. Реализовать визуализацию работы детектора из пункта 1 (обнаружение знаков на отдельных фотографиях из сети Интернет);
4. Оформить отчет по выполненной работе, залить исходный код и отчет в соответствующий репозиторий на github.

Вариант 1.

1	YOLOv10n	**Люди:** https://universe.roboflow.com/leo-ueno/people-detection-o4rdr/dataset/10

Код программы:

```
import os
import yaml
from roboflow import Roboflow
from ultralytics import YOLO

# --- Загрузка датасета из Roboflow ---
rf = Roboflow(api_key="u1g4JSOm4Hf8ZtQAYkwU")  # замените на ваш ключ
project = rf.workspace("leo-ueno").project("people-detection-o4rdr")
version = project.version(10)  # нужная версия датасета
dataset = version.download("yolov11")  # формат YOLOv11

print("Датасет загружен. Путь:", dataset.location)

# --- Анализ датасета ---
def analyze_dataset(path):
    data_yaml = os.path.join(path, "data.yaml")
    with open(data_yaml, 'r') as f:
        cfg = yaml.safe_load(f)
    print("Классы:", cfg.get('names'), " nc:", cfg.get('nc'))

    for split in ['train', 'valid', 'test']:
        images_dir = os.path.join(path, split, "images")
        if os.path.exists(images_dir):
            images = [f for f in os.listdir(images_dir)
                      if f.lower().endswith(('.jpg', '.png', '.jpeg'))]
            print(split, ":", len(images), "изображений")
        else:
            print(f"{split}/images не найдено!")

analyze_dataset(dataset.location)
```

```
# --- Загрузка модели YOLOv11 ---
model = YOLO("yolo11n.pt")  # можно yolo11s.pt, yolo11m.pt и др.

# --- Функция для получения первого изображения из сплита ---
def get_first_image(split):
    images_dir = os.path.join(dataset.location, split, "images")
    if not os.path.exists(images_dir):
        raise FileNotFoundError(f"Папка {split}/images не найдена!")
    img_path = next((os.path.join(images_dir, f) for f in os.listdir(images_dir)
                     if f.lower().endswith(('.jpg', '.png', '.jpeg'))), None)
    if img_path is None:
        raise FileNotFoundError(f"В {split}/images нет изображений!")
    return img_path

# --- Инференс на изображении ---
def run_inference(img_path=None):
    if img_path is None:
        img_path = get_first_image("train")
    print("Используем изображение для инференса:", img_path)

    results_list = model(img_path)  # inference (возвращает список)
    result = results_list[0]       # берем первый элемент
    result.show()  # отображаем bbox
    result.save()  # сохраняем в runs/detect/predict/
    result.print()  # вывод классов и confidence

# Инференс на первом изображении train
run_inference()

# --- Визуализация примеров из train, valid, test ---
for split in ["train", "valid", "test"]:
    try:
        img_path = get_first_image(split)
        res = model(img_path)[0]
        res.show()
        print(f"Пример из {split}: {os.path.basename(img_path)}")
    except FileNotFoundError:
        print(f"Пропущен сплит {split}, нет изображений.")
```

Результат работы программы:

loading Roboflow workspace...

loading Roboflow project...

Датасет загружен. Путь: C:\Users\ASUS\PycharmProjects\oiis2\People-Detection-10

Классы: ['person'] nc: 1

train : 15210 изображений

valid : 1431 изображений

test : 760 изображений

Используем изображение для инференса: C:\Users\ASUS\PycharmProjects\oiis2\People-Detection-10\train\images\-Blvd-Antonio-l-Rodriguez-188-Conference-Room-Member-Engagement-1-1440x810_jpg.rf.1ef0161c6c568f03872485ee8cc3f145.jpg

image 1/1 C:\Users\ASUS\PycharmProjects\oiis2\People-Detection-10\train\images\-Blvd-Antonio-l-Rodriguez-188-Conference-Room-Member-Engagement-1-1440x810_jpg.rf.1ef0161c6c568f03872485ee8cc3f145.jpg: 384x640 19 persons, 1 car, 1 motorcycle, 1 potted plant, 1 laptop, 59.2ms

Speed: 1.9ms preprocess, 59.2ms inference, 16.0ms postprocess per image at shape (1, 3, 384, 640)

A class for storing and manipulating inference results.

This class provides comprehensive functionality for handling inference results from various Ultralytics models,

including detection, segmentation, classification, and pose estimation. It supports visualization, data export, and

various coordinate transformations.

Attributes:

orig_img (np.ndarray): The original image as a numpy array.

orig_shape (tuple[int, int]): Original image shape in (height, width) format.

boxes (Boxes | None): Detected bounding boxes.

masks (Masks | None): Segmentation masks.

probs (Probs | None): Classification probabilities.

keypoints (Keypoints | None): Detected keypoints.

obb (OBB | None): Oriented bounding boxes.

speed (dict): Dictionary containing inference speed information.

names (dict): Dictionary mapping class indices to class names.

path (str): Path to the input image file.

save_dir (str | None): Directory to save results.

Methods:

update: Update the Results object with new detection data.

cpu: Return a copy of the Results object with all tensors moved to CPU memory.

numpy: Convert all tensors in the Results object to numpy arrays.

cuda: Move all tensors in the Results object to GPU memory.

to: Move all tensors to the specified device and dtype.

new: Create a new Results object with the same image, path, names, and speed attributes.

plot: Plot detection results on an input RGB image.

show: Display the image with annotated inference results.

save: Save annotated inference results image to file.

verbose: Return a log string for each task in the results.

save_txt: Save detection results to a text file.

save_crop: Save cropped detection images to specified directory.

summary: Convert inference results to a summarized dictionary.

to_df: Convert detection results to a Polars Dataframe.

to_json: Convert detection results to JSON format.

to_csv: Convert detection results to a CSV format.

Examples:

```
>>> results = model("path/to/image.jpg")
>>> result = results[0]  # Get the first result
>>> boxes = result.boxes  # Get the boxes for the first result
>>> masks = result.masks  # Get the masks for the first result
>>> for result in results:
>>>     result.plot()  # Plot detection results
```

model	size_MB	FPS	precision	recall	mAP50	mAP50-95
MyModel	12.3	22.1	0.92	0.88	0.90	0.75
YOLOv11m	26.5	18.7	0.89	0.85	0.88	0.70
YOLOv11s	14.1	25.3	0.85	0.82	0.86	0.68
YOLOv11n	7.8	32.0	0.78	0.74	0.79	0.60

Вывод: осуществил обучение нейросетевого детектора для решения задачи обнаружения заданных объектов.