Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра интеллектуально-информационных технологий

Лабораторная работа №3
По дисциплине «Обработка изображений в ИС»

Тема: «Обучение детекторов объектов»

Выполнила:
студентка 4 курса
группы ИИ-24
Коцуба Е.М.
Проверила:
Андренко К.В.

Брест 2025

Цель работы: осуществлять обучение нейросетевого детектора для решения задачи обнаружения заданных объектов.

Вариант 5

В-т	**Детектор**	**Датасет**
5	YOLOv10 m	**Счетчики расхода воды:** https://universe.roboflow.com/koer3741-gmail-com/watermeteramrv2/dataset/1

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.

Код программы:

```
from roboflow import Roboflow
import os
import requests
from io import BytesIO

rf = Roboflow(api_key="2OFRWtLUJqdrNuwZhhPk")
project = rf.workspace("koer3741-gmail-
com").project("watermeteramrv2")
dataset = project.version(1).download("yolov8")
dataset_path = dataset.location
data_yaml_path = os.path.join(dataset_path, "data.yaml")
with open(data_yaml_path, 'r') as f:
    print("Содержимое data.yaml:")
    print(f.read())
train_images = len(os.listdir(os.path.join(dataset_path,
"train", "images")))
```

```
val_images = len(os.listdir(os.path.join(dataset_path, "valid",
"images")))
test_images = len(os.listdir(os.path.join(dataset_path, "test",
"images"))) if os.path.exists(os.path.join(dataset_path,
"test")) else 0
print(f"\nРазмеры датасета:")
print(f"Train: {train_images} изображений")
print(f"Valid: {val_images} изображений")
print(f"Test: {test_images} изображений")
print("\nКлассы:
'0','1','2','3','4','5','6','7','8','9','counter','liters'")

from ultralytics import YOLO

model = YOLO("yolov10m.pt")
results = model.train(
    data=data_yaml_path,
    epochs=10,
    imgsz=640,
    batch=16,
    name="yolov10m_water_meter_detection",
    device=0
)

metrics = model.val(data=data_yaml_path)
print("Метрики валидации:")
print(f"mAP@0.5: {metrics.box.map50:.4f}")
print(f"mAP@0.5:0.95: {metrics.box.map:.4f}")
print("\nПолные метрики:")
print(metrics)

import supervision as sv
from PIL import Image
import matplotlib.pyplot as plt
import requests
from io import BytesIO

internet_image_url =
"https://images.prom.ua/1879407902_w640_h640_1879407902.jpg"
response = requests.get(internet_image_url)
img = Image.open(BytesIO(response.content))
img.save("internet_water_meter.jpg")
test_image_path = "internet_water_meter.jpg"
```

```
results = model(test_image_path)

plt.figure(figsize=(10, 10))
plt.imshow(img)
plt.axis('off')

for result in results:
    boxes = result.boxes
    if boxes is not None:
        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()
            conf = box.conf[0].cpu().numpy()
            cls = int(box.cls[0].cpu().numpy())
            class_name = model.names[cls]
            label = f"{class_name} {conf:.2f}"
            plt.gca().add_patch(plt.Rectangle((x1, y1), x2-x1,
y2-y1, fill=False, color='red', linewidth=2))
            plt.text(x1, y1-10, label, color='red', fontsize=12,
bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

plt.title("Визуализация детекций на изображении из интернета")
plt.show()
result.save("detection_result.jpg")
print("Результат сохранен как detection_result.jpg")
```

Результат работы программы:

Ключевые характеристики датасета

- Классы: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'counter', 'liters' (nc=12).
- Размеры:
  - Train: 3478 изображений.
  - Valid: 389 изображений.
  - Test: 389 изображений.
- Датасет загружен из Roboflow, формат YOLOv8 (совместим с YOLOv10).

Процесс обучения

- Модель: YOLOv10m (medium-версия, ~16.5 млн параметров, 64 GFLOPs).
- Параметры: imgsz=640, batch=16, optimizer=AdamW (авто-подбор lr=0.000625), device=GPU (Tesla T4).

- Augmentation: Стандартные (mosaic, flip, etc.), но mosaic отключён после 0 эпох (close_mosaic=10, но epochs=10).
- Время обучения: ~0.388 часа (~23 минуты) на GPU.
- Прогресс метрик по эпохам

Прогресс метрик по эпохам:

Epoch	box_loss	cls_loss	dfl_loss	mAP@0.5	mAP@0.5:0.95
1	3.031	6.059	2.57	0.630	0.299
2	2.933	2.280	2.42	0.644	0.320
3	2.913	1.879	2.419	0.741	0.392
4	2.837	1.667	2.39	0.876	0.486
5	2.785	1.490	2.36	0.903	0.473
6	2.725	1.351	2.32	0.912	0.497
7	2.653	1.242	2.27	0.937	0.553
8	2.586	1.167	2.235	0.954	0.566
9	2.531	1.088	2.219	0.956	0.587
10	2.447	1.014	2.182	0.960	0.587

Визуализация детекций на изображении из интернета

Вывод: модель хорошо детектирует 'counter' (mAP=0.783) — ключевой элемент. Цифры имеют mAP ~0.49–0.61, что нормально для мелких объектов. Общий mAP 0.587 указывает на среднюю точность (возможно, улучшить больше эпохами или аугментацией). Классы с меньшим количеством экземпляров (напр., '8', '9') имеют ниже mAP.