Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «ОИвИС»

Тема: «Обучение детекторов объектов»

Выполнила:
Студентка 4 курса
Группы ИИ-23
Тутина Е.Д.
Проверила:
Андренко К. В.

Брест 2025

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

Вариант 11

11	YOLOv11m	**Счетчики расхода воды:** https://universe.roboflow.com/koer3741-gmail-com/watermeteramrv2/dataset/1

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.

Код работы:

```
!pip install -U pip

!pip install ultralytics roboflow supervision

from google.colab import drive

drive.mount('/content/drive')

!mkdir -p /content/drive/MyDrive/watermeter_project

!unzip -q /WaterMeterAMRV2.v1i.yolov11.zip -d /content/watermeter_dataset

!ls -la /content/watermeter_dataset

!sed -n '1,200p' /content/watermeter_dataset/data.yaml  || true

\

import yaml, os

def find_data_yaml(root="/content"):
```

```
    for dirpath, dirnames, filenames in os.walk(root):
        if "data.yaml" in filenames:
            return os.path.join(dirpath, "data.yaml")
    return None

data_yaml_path = find_data_yaml("/content")
print("Found data.yaml:", data_yaml_path)

if data_yaml_path:
    with open(data_yaml_path, "r") as f:
        print(f.read())
else:
    print("data.yaml не найден — укажи путь вручную")



path_to_data_yaml = data_yaml_path or "/content/WaterMeterAMRV2.v1i.yolov11/data.yaml"
from ultralytics import YOLO
import os, sys

model_name_primary = "yolo11m.pt"
model_name_fallback = "yolo11n.pt"

try:
    model = YOLO(model_name_primary)
    print("Loaded model:", model_name_primary)
except Exception as e:
        print(f"Не удалось загрузить {model_name_primary}: {e}\nПопробуем {model_name_fallback}...")
    model = YOLO(model_name_fallback)
    print("Loaded model:", model_name_fallback)
```

```
EPOCHS = 50

BATCH = 16

IMGSZ = 640

EXPERIMENT_NAME = "watermeter_yolo11m"


results = model.train(data=path_to_data_yaml,

                      epochs=EPOCHS,

                      imgsz=IMGSZ,

                      batch=BATCH,

                      name=EXPERIMENT_NAME,

                      project="/content/drive/MyDrive/watermeter_project")
print("Training finished. Results object:", results)
```

Результаты обучения модели:

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

 1/50      7.93G      1.447      1.746      1.208         55        640:

100% ━━━━━━━━━━━━━━━━━━━━ 218/218 1.6it/s 2:17

                 Class      Images  Instances       Box(P          R      mAP50

mAP50-95): 100% ━━━━━━━━━━━━━━━━━━━━ 13/13 1.2it/s 10.5s

                   all         389        2059      0.769      0.688      0.754

0.395

Epoch    GPU_mem   box_loss  cls_loss    dfl_loss   Instances      Size

50/50     8.12G     0.9779    0.3892       1.009       33            640:

100% ━━━━━━━━━━━━━━━━━━━━ 218/218 1.7it/s 2:05

        Class Images Instances Box(P R mAP50 mAP50-95):

100% ━━━━━━━━━━━━━━━━━━━━ 13/13 2.0it/s 6.5s all 389 2059 0.959 0.969 0.984 0.642
```

```
from ultralytics import YOLO

path_to_data_yaml                =                data_yaml_path                or

"/content/WaterMeterAMRV2.v1i.yolov11/data.yaml"
```

```
weights_path = f"/content/drive/MyDrive/watermeter_yolo11m2/weights/best.pt"

import os

if not os.path.exists(weights_path):

    import glob

                cand    =    glob.glob("/content/runs/train/*/weights/best.pt")    +
glob.glob("/content/drive/MyDrive/watermeter_project/*/weights/best.pt")

    weights_path = cand[0] if cand else None


print("Weights to validate:", weights_path)


if weights_path:

    model = YOLO(weights_path)

      metrics  =  model.val(data=path_to_data_yaml,  split="test")   # split может
быть 'val' или 'test' в data.yaml

    print("Validation metrics:", metrics)

else:

       print("Веса  не  найдены  —  убедись,  что  обучение  завершилось  и  best.pt
существует")
```

```
esults_dict:  {'metrics/precision(B)':  0.9654852401486472,  'metrics/recall(B)':  0.9511744219172167,
'metrics/mAP50(B)':    0.9734940751706667,    'metrics/mAP50-95(B)':    0.615887498245573,    'fitness':
0.615887498245573}

save_dir: PosixPath('/content/runs/detect/val2')

speed:    {'preprocess':    1.6145884267361532,    'inference':    24.123917282776077,    'loss':
0.0007607095113460972, 'postprocess': 1.0730754061702}

stats: {'tp': [], 'conf': [], 'pred_cls': [], 'target_cls': [], 'target_img': []}
```

```
from ultralytics import YOLO

import cv2

import numpy as np

from PIL import Image

import os

from IPython.display import display, Image as IPyImage
```

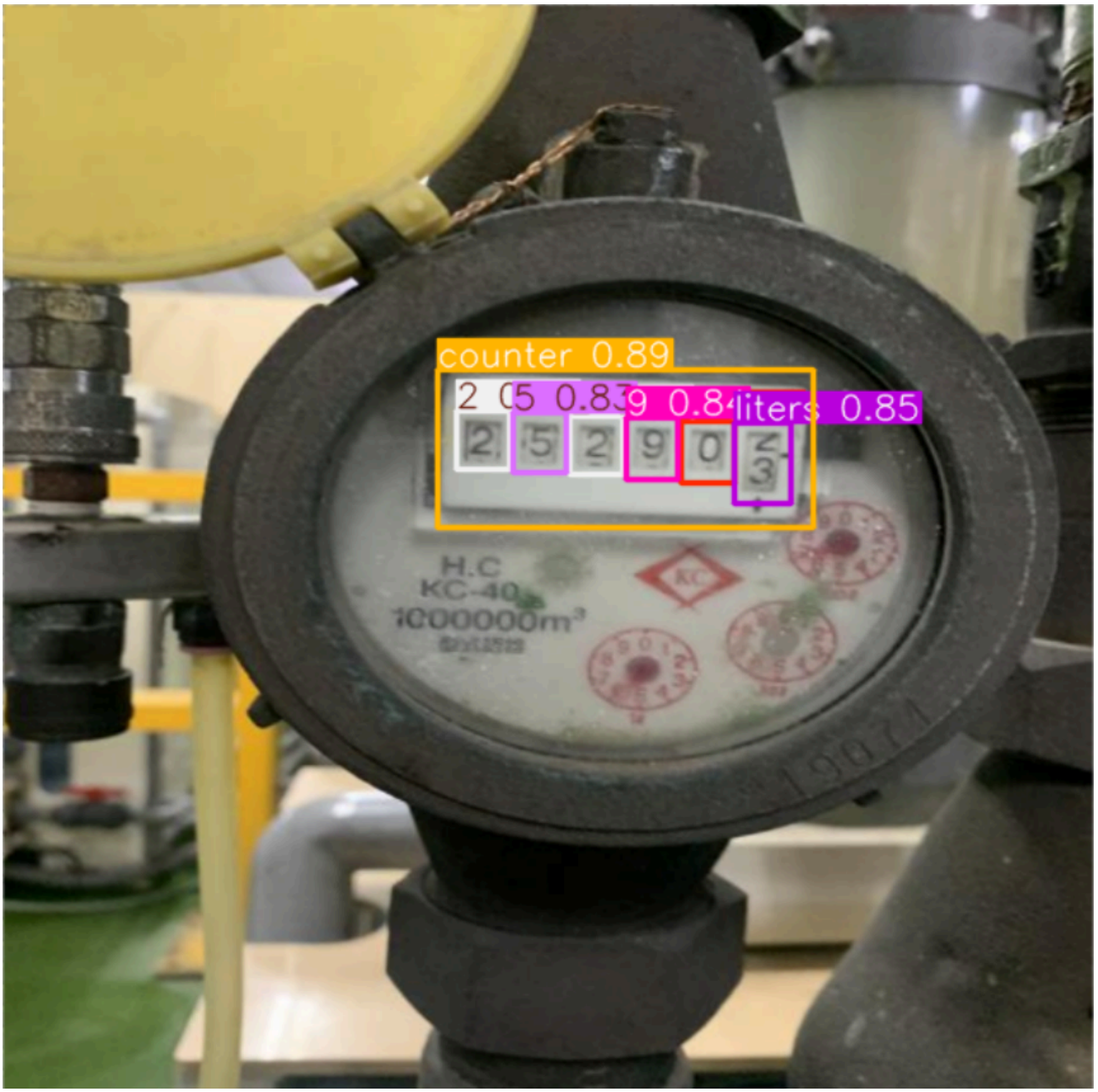
```
if weights_path and os.path.exists(weights_path):
    model = YOLO(weights_path)
else:
    model = YOLO(model_name_fallback)
    print("Используем fallback модель для демонстрации")


def load_image_from_path(image_path):
    """Загружает изображение с локального пути"""
    if not os.path.exists(image_path):
        raise FileNotFoundError(f"Файл не найден: {image_path}")


    img = Image.open(image_path).convert("RGB")
    return np.array(img)  # RGB


def predict_and_show(image_path, conf=0.1, imgsz=640,
save_path="/content/pred_result.jpg"):
    """Предсказание и отображение результата для локального изображения"""
    img = load_image_from_path(image_path)
    img = cv2.resize(img, (imgsz, imgsz))  # масштабируем
    results = model.predict(source=img, imgsz=imgsz, conf=conf)
    annotated = results[0].plot()
    display(Image.fromarray(annotated))
    print(results[0].boxes.data)


test_photo = "/test_photo.png"
predict_and_show(test_photo, conf=0.3, imgsz=640,
save_path="/content/drive/MyDrive/watermeter_project/pred_example.jpg")
print("Используемые веса:", weights_path)
```




```
tensor([[254.9146, 215.6295, 473.0639, 307.1847,   0.8909,  10.0000],
        [427.2786, 246.2912, 460.4287, 293.1766,   0.8452,  11.0000],
        [364.3181, 243.3836, 394.9291, 279.5813,   0.8427,   9.0000],
        [298.1723, 240.5833, 329.5530, 275.9720,   0.8295,   5.0000],
        [265.3412, 239.6604, 296.0730, 273.4600,   0.8293,   2.0000],
        [331.3122, 242.6616, 362.3577, 276.7952,   0.8153,   2.0000],
        [396.9951, 245.7336, 427.2407, 281.2375,   0.7793,   0.0000]], device='cuda:0')
```

```
Используемые                                                    веса:
/content/drive/MyDrive/watermeter_project/watermeter_yolo11m2/weights/best.pt
```

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