

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

Обработка изображений в ИС
Лабораторная работа №3
Обучение детекторов объектов

Выполнил:
Студент 4-го курса
Группы ИИ-24
Поддубный Ю. А.
Проверила:
Андренко К. В.

Брест-2025

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

Задание по вариантам:

№ варианта	Детектор	Датасет
14	YOLOv12n	Номерные знаки авто: https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11

Код программы:

download_dataset.py

```
import os
from roboflow import Roboflow

ROBOFLOW_API_KEY = os.getenv("ROBOFLOW_API_KEY",
"<BAШ_API_KEY>")
WORKSPACE = "roboflow-universe-projects"
PROJECT = "license-plate-recognition-rxg4e"
VERSION = 11
OUT_DIR = "./dataset"

def main():

    print("Connecting to Roboflow...")
    rf = Roboflow(api_key=ROBOFLOW_API_KEY)
```

```

print("Loading workspace and project...")
project = rf.workspace(WORKSPACE).project(PROJECT)

print(f"Downloading dataset version {VERSION} ...")
version = project.version(VERSION)
version.download("yolov5", location=OUT_DIR) #
print("Dataset downloaded successfully!")

if __name__ == "__main__":
    main()

```

prepare_data.py

```

import os
import yaml

DATA_ROOT = "./dataset"
OUT = os.path.join(DATA_ROOT, "data.yaml")

def infer_classes_from_names_file(data_root):
    names_path = os.path.join(data_root, "data.json")
    if os.path.exists(names_path):
        import json
        d = json.load(open(names_path))
        classes = d.get("names") or d.get("labels") or []
        return classes
    return ["license_plate"]

def main():
    classes = infer_classes_from_names_file(DATA_ROOT)
    data = {
        "names": classes,
        "nc": len(classes),
        "train": os.path.join(DATA_ROOT, "train"),
        "val": os.path.join(DATA_ROOT, "valid"),
        "test": os.path.join(DATA_ROOT, "test")
    }
    with open(OUT, "w") as f:
        yaml.dump(data, f, sort_keys=False)
    print("Wrote", OUT)
    print("classes:", classes)

if __name__ == "__main__":
    main()

```

train.py

```
from ultralytics import YOLO

DATA_YAML = "./dataset/data.yaml"
OUTPUT_DIR = "./runs/train/license_plate_yolov12n"

def main():
    model = YOLO("yolov8n.pt")

    results = model.train(
        data=DATA_YAML,
        epochs=100,
        imgsz=704,
        batch=32,
        save=True,
        name="license_plate_yolov12n",
        project="runs/train",
        device=0,
        workers=8,
        patience=20,
        optimizer="AdamW",
        lr0=0.001,
        warmup_epochs=3,
        cos_lr=True,
    )

    print("Training finished. Results:", results)

if __name__ == "__main__":
    main()
```

evaluate.py

```
from ultralytics import YOLO

WEIGHTS =
"./runs/train/license_plate_yolov12n12/weights/best.pt"
DATA_YAML = "./dataset/data.yaml"

def main():
    model = YOLO(WEIGHTS)
    metrics = model.val(data=DATA_YAML, batch=16, imgsz=704)
    print("Validation metrics:", metrics)

if __name__ == "__main__":
    main()
```

visualize.py

```
import os
from ultralytics import YOLO
from PIL import Image, ImageDraw

WEIGHTS = "runs/train/license_plate_yolov12n/weights/best.pt"
IMG_DIR = "./images_in"
OUT_DIR = "./images_out"

os.makedirs(OUT_DIR, exist_ok=True)

def draw_boxes_and_save(image_path, results, save_path):
    img = Image.open(image_path).convert("RGB")
    draw = ImageDraw.Draw(img)

    boxes = results.boxes.xyxy.cpu().numpy()
    confs = results.boxes.conf.cpu().numpy()
    classes = results.boxes.cls.cpu().numpy()

    for (x1, y1, x2, y2), conf, cls_id in zip(boxes, confs,
classes):
        draw.rectangle([(x1, y1), (x2, y2)], outline="red",
width=3)
        text = f"{results.names[int(cls_id)]} {conf:.2f}"
        draw.text((x1 + 5, y1 + 5), text, fill="red")

    img.save(save_path)

def main():
    model = YOLO(WEIGHTS)

    images = [
        f for f in os.listdir(IMG_DIR)
        if f.lower().endswith((".jpg", ".jpeg", ".png"))
    ]

    if not images:
        print(f"В папке {IMG_DIR} нет изображений!")
        return

    for i, img_file in enumerate(images):
        img_path = os.path.join(IMG_DIR, img_file)
        out_path = os.path.join(OUT_DIR, f"pred_{img_file}")
```

```

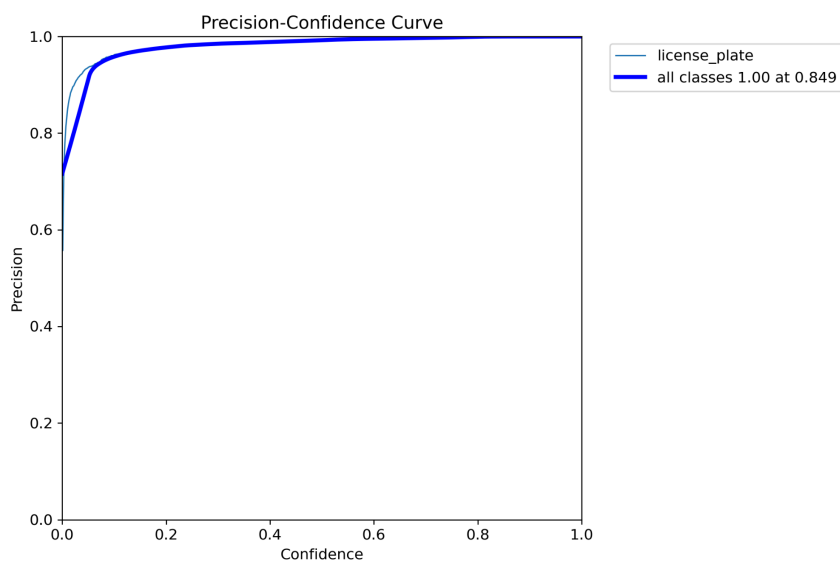
        preds = model.predict(img_path, imgsz=1280, conf=0.25,
save=False)
        res = preds[0]
        draw_boxes_and_save(img_path, res, out_path)
        print(f"Saved: {out_path}")

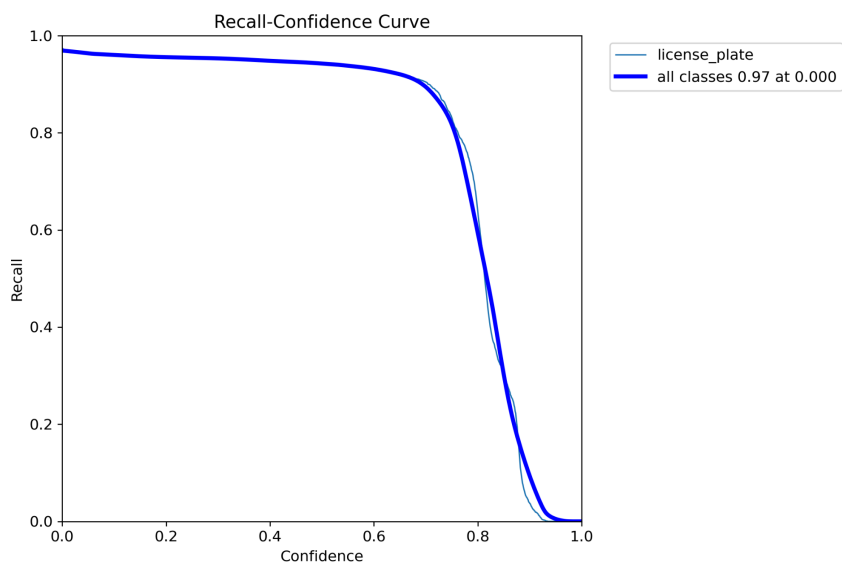
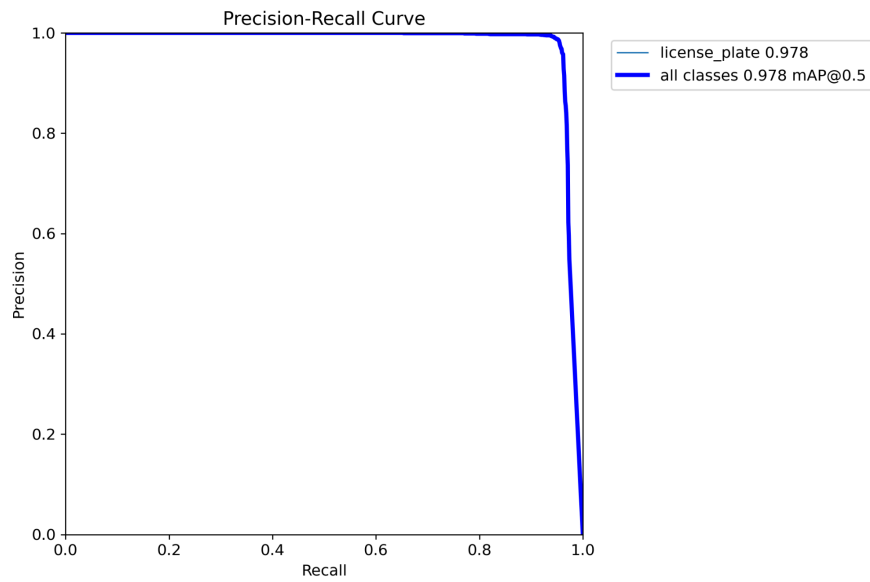
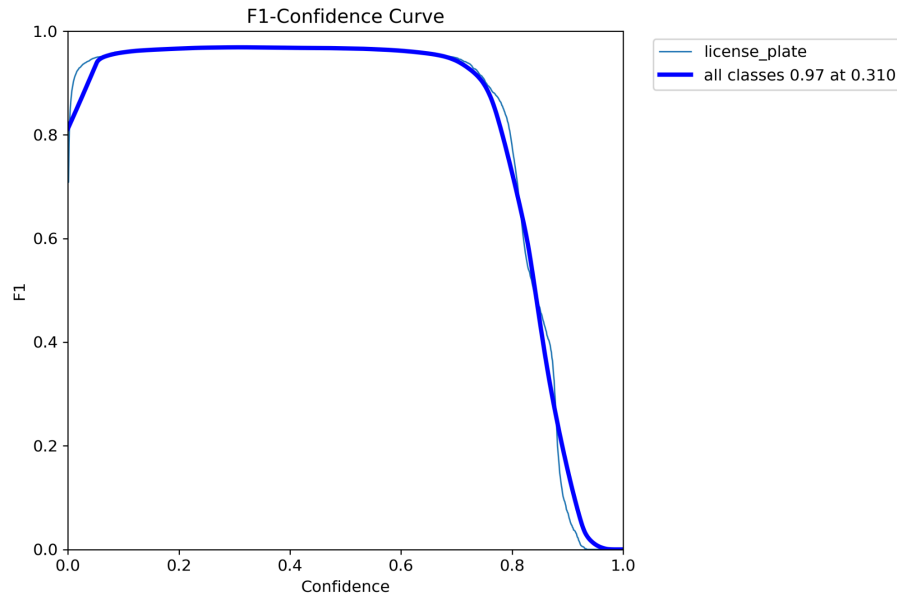
if __name__ == "__main__":
    main()

```

Результат работы программы:

Метрика	Значение
Precision (P)	0.985
Recall (R)	0.953
mAP@0.5	0.978
mAP@0.5–0.95	0.723
Изображений (val)	2048
Объектов (instances)	2195







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