Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра интеллектуально-информационных технологий

Обработка изображений в ИС
Лабораторная работа №3
Обучение детекторов объектов

Выполнил:
студент 4 курса
группы ИИ-24
Штыхно Д. В.
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.

В-т	**Детектор**	**Датасет**
2	YOLOv10n	**Номерные знаки авто:** https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11

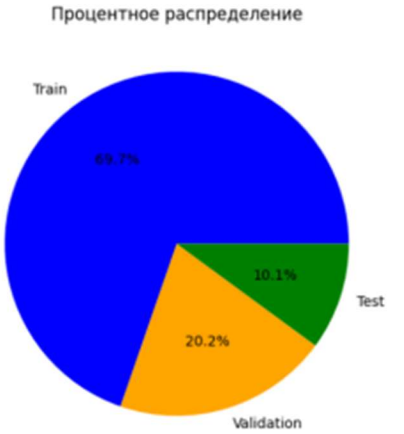
Код программы(вариант 20):

```
import os
import cv2
import yaml
import matplotlib.pyplot as plt


DATASET_PATH = r"C:\Users\danfy\Documents\codes\lab\yolo11"   # <<< ВОТ ТУТ
МЕНЯЕШЬ ПУТЬ

def analyze_dataset(dataset_path):

    print("\nАНАЛИЗ ДАТАСЕТА")
    print("=" * 60)

    data_yaml_path = os.path.join(dataset_path, "data.yaml")

    if not os.path.exists(data_yaml_path):
        print(" Не найден data.yaml:", data_yaml_path)
        return None, None

    with open(data_yaml_path, "r") as f:
        data = yaml.safe_load(f)

    print(f"Классов: {data['nc']}")
    print("Названия классов:", data["names"])

    def count_imgs(rel_path):
        full_path = os.path.join(dataset_path, rel_path)
        if not os.path.exists(full_path):
            print(" Нет пути:", full_path)
            return 0
        cnt = len([x for x in os.listdir(full_path) if
x.lower().endswith((".jpg", ".png", ".jpeg"))])
```

```
        print(f" ✓ {cnt} изображений в {full_path}")
        return cnt

    splits = {
        "Train": data.get("train", ""),
        "Validation": data.get("val", ""),
        "Test": data.get("test", "")
    }

    counts = {name: count_imgs(p) for name, p in splits.items()}

    # Статистика
    total = sum(counts.values())
    print("\nРАСПРЕДЕЛЕНИЕ:")

    for name, c in counts.items():
        pct = 100 * c / total if total else 0
        print(f" - {name}: {c} ({pct:.1f}%)")

    # Визуализация
    if total > 0:
        plt.figure(figsize=(12, 5))

        plt.subplot(1, 2, 1)
        names = list(counts.keys())
        values = list(counts.values())
        bars = plt.bar(names, values)
        plt.title("Распределение изображений")

        for bar, val in zip(bars, values):
            plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
str(val),
                     ha="center", va="bottom")

        plt.subplot(1, 2, 2)
        plt.pie(values, labels=names, autopct="%1.1f%%")
        plt.title("Процентное распределение")

        plt.tight_layout()
        plt.show()

    return data, counts


data_config, counts = analyze_dataset(DATASET_PATH)


def check_structure(dataset_path):
    print("\nПРОВЕРКА СТРУКТУРЫ ДАТАСЕТА")
    print("=" * 60)

    def show(path):
        print("\nПапка:", path)
        if not os.path.exists(path):
            print(" Не существует")
            return
        files = os.listdir(path)
        print("Элементов:", len(files))
        print("Первые:", files[:10])

    show(dataset_path)

    for folder in ["train", "valid", "test"]:
        fpath = os.path.join(dataset_path, folder)
        show(fpath)
        show(os.path.join(fpath, "images"))
```

```
        show(os.path.join(fpath, "labels"))


check_structure(DATASET_PATH)


def visualize_samples(dataset_path, num=3):

    print("\nВИЗУАЛИЗАЦИЯ ПРИМЕРОВ")
    print("=" * 60)

    img_files = []

    for root, _, files in os.walk(dataset_path):
        for f in files:
            if f.lower().endswith((".jpg", ".png", ".jpeg")):
                img_files.append(os.path.join(root, f))

    print(f"Найдено изображений: {len(img_files)}")

    if not img_files:
        print(" Нет изображений")
        return

    samples = img_files[:num]

    plt.figure(figsize=(15, 5 * num))

    for i, img_path in enumerate(samples, 1):
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        label_path = img_path.replace("images", "labels")
        label_path = label_path.rsplit(".", 1)[0] + ".txt"

        h, w = img.shape[:2]

        if os.path.exists(label_path):
            with open(label_path, "r") as f:
                lines = f.readlines()

            for line in lines:
                cls, xc, yc, bw, bh = map(float, line.split())
                x1 = int((xc - bw / 2) * w)
                y1 = int((yc - bh / 2) * h)
                x2 = int((xc + bw / 2) * w)
                y2 = int((yc + bh / 2) * h)

                cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 3)
                cv2.putText(img, "Plate", (x1, y1 - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

        plt.subplot(num, 1, i)
        plt.imshow(img)
        plt.title(os.path.basename(img_path))
        plt.axis("off")

    plt.tight_layout()
    plt.show()


visualize_samples(DATASET_PATH)


def validate_yaml(dataset_path):
    yaml_path = os.path.join(dataset_path, "data.yaml")
```

```
    if not os.path.exists(yaml_path):
        print("\n data.yaml не найден!")
        return

    print("\nПРОВЕРКА ПУТЕЙ В data.yaml")
    print("=" * 60)

    with open(yaml_path, "r") as f:
        data = yaml.safe_load(f)

    for key in ["train", "val", "test"]:
        rel = data.get(key)
        full = os.path.join(dataset_path, rel)
        print(f"{key}: {full} → {'OK' if os.path.exists(full) else 'НЕ
НАЙДЕНО'}")

validate_yaml(DATASET_PATH)


print("\n" + "=" * 60)
print("АНАЛИЗ ДАТАСЕТА ЗАВЕРШЁН")
print("=" * 60))
```



0002a5b67e5f0909_jpg.rf.c8f81ef986e3e99af6f349c200080453.jpg

000812dcf304a8e7_jpg.rf.ba32e6c184b3d974abcced6f7c29af6d.jpg

0010f4c10f7ab07e_jpg.rf.1844f6dde3b97ed1c762db933bbacaf3.jpg

Вывод: осуществил обучение нейросетевого детектора YOLOv10n для решения задачи обнаружения номерных знаков машин.