

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

**Лабораторная работа №3**

По дисциплине «Обработка изображений в интеллектуальных системах»

Тема: «Обучение детекторов объектов»

**Выполнила:**

Студентка 4 курса

Группы ИИ-24

Максимович А. И.

**Проверила:**

Андренко К. В.

Брест 2025

**Цель:** 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**

№ в-а	Дететктор	Датасет
11	YOLOv11m	<b>Счетчики расхода воды:</b> <a href="https://universe.roboflow.com/koer3741-gmail-com/watermeteramrv2/dataset/1">https://universe.roboflow.com/koer3741-gmail-com/watermeteramrv2/dataset/1</a>

### **Код:**

```
import os
import shutil
import cv2
from ultralytics import YOLO
import yaml
import random
import torch
import torchvision

# ===== КОНФИГУРАЦИЯ =====
DOWNLOADED_DATASET_PATH = 'watermeteramrv2'
MODEL_TYPE = 'yolo11m.pt'
EPOCHS = 10
IMGSZ = 416
BATCH = 16
OPTIMIZER = 'SGD'
MODEL_NAME = 'yolo11m'
YOLO_DATASET = 'watermeteramrv2_res'
TEST_IMAGES = ["20220216_222607_jpg.rf.1483f40dac6a642b5bad42a24beeb52a.jpg",
"egohands-public-1620849869759_png_jpg.rf.c537eb75df29be09ebea80d01e0b4c76.jpg"]
# =====
```

```

def check_gpu_availability():
    """Проверка доступности GPU и вывод информации"""
    print("=" * 50)
    print("ПРОВЕРКА ДОСТУПНОСТИ GPU")
    print("=" * 50)

    mps_available = torch.backends.mps.is_available()
    cuda_available = torch.cuda.is_available()

    print(f"MPS (Mac) доступен: {mps_available}")
    print(f"CUDA доступен: {cuda_available}")

    if mps_available:
        device = torch.device('mps')
        print(f"Используется устройство: {device} (MPS для Mac)")
    elif cuda_available:
        device = torch.device('cuda')
        print(f"Используется устройство: {device} (CUDA)")
    else:
        device = torch.device('cpu')
        print("ВНИМАНИЕ: GPU не доступен, обучение будет на CPU!")

    print()
    return device

def setup_dataset_from_download():
    """Настройка датасета, скачанного вручную с RoboFlow"""
    print("Настройка скачанного датасета...")

    if not os.path.exists(DOWNLOADED_DATASET_PATH):
        raise FileNotFoundError(f"Папка с датасетом не найдена: {DOWNLOADED_DATASET_PATH}")

    # Читаем конфигурационный файл датасета
    original_data_yaml_path = os.path.join(DOWNLOADED_DATASET_PATH,
    'data.yaml')
    with open(original_data_yaml_path, 'r') as f:
        data_config = yaml.safe_load(f)

    # Создаем структуру папок для YOLO
    yolo_train_img_dir = os.path.join(YOLO_DATASET, 'train', 'images')
    yolo_train_lbl_dir = os.path.join(YOLO_DATASET, 'train', 'labels')
    yolo_val_img_dir = os.path.join(YOLO_DATASET, 'valid', 'images')
    yolo_val_lbl_dir = os.path.join(YOLO_DATASET, 'valid', 'labels')
    yolo_test_img_dir = os.path.join(YOLO_DATASET, 'test', 'images')
    yolo_test_lbl_dir = os.path.join(YOLO_DATASET, 'test', 'labels')

    for dir_path in [yolo_train_img_dir, yolo_train_lbl_dir,
                    yolo_val_img_dir, yolo_val_lbl_dir,
                    yolo_test_img_dir, yolo_test_lbl_dir]:
        os.makedirs(dir_path, exist_ok=True)

    # Функция для копирования данных с правильным определением путей
    def copy_split(split_name, yolo_img_dir, yolo_lbl_dir):
        print(f"\nКопирование {split_name} данных...")

        # Определяем пути к исходным данным
        # RoboFlow датасеты обычно имеют структуру: dataset/train,
        dataset/valid, dataset/test
        # или пути указаны в data.yaml

```

```

        # Вариант 1: Проверяем стандартную структуру RoboFlow
        original_img_dir = os.path.join(DOWNLOADED_DATASET_PATH, split_name,
'images')
        original_lbl_dir = os.path.join(DOWNLOADED_DATASET_PATH, split_name,
'labels')

        # Вариант 2: Если структура другая, используем пути из data.yaml
        if not os.path.exists(original_img_dir) and split_name in
data_config:
            original_path = data_config[split_name]
            if original_path.startswith('../'):
                # Убираем ../ из пути
                original_path = original_path[3:]
            original_img_dir = os.path.join(DOWNLOADED_DATASET_PATH,
original_path)
            original_lbl_dir = original_img_dir.replace('images', 'labels')

        print(f"Источник изображений: {original_img_dir}")
        print(f"Источник разметок: {original_lbl_dir}")

        if os.path.exists(original_img_dir):
            # Копируем изображения
            image_files = [f for f in os.listdir(original_img_dir) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
            print(f"Найдено {len(image_files)} изображений")

            for img_file in image_files:
                src_img = os.path.join(original_img_dir, img_file)
                dst_img = os.path.join(yolo_img_dir, img_file)
                shutil.copy2(src_img, dst_img)

                # Копируем соответствующие файлы разметки
                lbl_file = os.path.splitext(img_file)[0] + '.txt'
                src_lbl = os.path.join(original_lbl_dir, lbl_file)
                dst_lbl = os.path.join(yolo_lbl_dir, lbl_file)
                if os.path.exists(src_lbl):
                    shutil.copy2(src_lbl, dst_lbl)
                else:
                    print(f"Предупреждение: файл разметки {src_lbl} не
найден")

            print(f"Скопировано {len(image_files)} изображений для
{split_name}")
        else:
            print(f"ОШИБКА: Папка {original_img_dir} не найдена!")
            # Создаем пустые папки чтобы избежать ошибок
            open(os.path.join(yolo_img_dir, '.keep'), 'w').close()
            open(os.path.join(yolo_lbl_dir, '.keep'), 'w').close()

        # Копируем данные
        copy_split('train', yolo_train_img_dir, yolo_train_lbl_dir)
        copy_split('val', yolo_val_img_dir, yolo_val_lbl_dir)
        copy_split('test', yolo_test_img_dir, yolo_test_lbl_dir)

        # Создаем обновленный data.yaml файл
        updated_data_yaml = {
            'path': os.path.abspath(YOLO_DATASET),
            'train': 'train/images',
            'val': 'valid/images',
            'test': 'test/images',

```

```

        'nc': data_config['nc'],
        'names': data_config['names']
    }

    yolo_data_yaml_path = os.path.join(YOLO_DATASET, 'data.yaml')
    with open(yolo_data_yaml_path, 'w') as f:
        yaml.dump(updated_data_yaml, f, default_flow_style=False)

    print("\nНастройка датасета завершена!")
    print(f"Количество классов: {data_config['nc']}")
    print(f"Классы: {data_config['names']}")

    # Проверяем, есть ли данные в папках
    for split in ['train', 'valid', 'test']:
        img_dir = os.path.join(YOLO_DATASET, split, 'images')
        if os.path.exists(img_dir):
            images = [f for f in os.listdir(img_dir) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
            print(f"Изображений в {split}: {len(images)}")

    return data_config['nc'], data_config['names'], yolo_data_yaml_path

def train_yolo_model(num_classes, class_names, data_yaml_path, device):
    print(f"\nОбучение модели YOLO для {num_classes} классов:
{class_names}...")
    print(f"Обучение на устройстве: {device}")

    # Отключаем AMP для MPS
    amp_enabled = device.type != 'mps'

    last_weights = f"./runs/detect/{MODEL_NAME}/weights/last.pt"
    best_weights = f"./runs/detect/{MODEL_NAME}/weights/best.pt"

    # Сначала проверяем существование data.yaml и данных
    if not os.path.exists(data_yaml_path):
        raise FileNotFoundError(f"Файл data.yaml не найден:
{data_yaml_path}")

    with open(data_yaml_path, 'r') as f:
        data_config = yaml.safe_load(f)

    # Проверяем существование папок с данными
    for split in ['train', 'val']:
        split_path = data_config.get(split, '')
        if split_path:
            full_path = os.path.join(data_config['path'], split_path) if not
os.path.isabs(split_path) else split_path
            if not os.path.exists(full_path):
                print(f"ПРЕДУПРЕЖДЕНИЕ: Папка {full_path} не найдена!")
            else:
                images = [f for f in os.listdir(full_path) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
                print(f"Найдено {len(images)} изображений в {full_path}")

    if os.path.exists(last_weights):
        print(f"Найдены предыдущие веса: {last_weights}, продолжаем
обучение")
        model = YOLO(last_weights)
    elif os.path.exists(best_weights):
        print(f"Найдены лучшие веса: {best_weights}, дообучаем с них")

```

```

        model = YOLO(best_weights)
    else:
        print(f"Начинаем обучение с нуля: {MODEL_TYPE}")
        model = YOLO(MODEL_TYPE)

    try:
        results = model.train(
            data=data_yaml_path,
            epochs=EPOCHS,
            imgsz=IMGSZ,
            batch=BATCH,
            name=MODEL_NAME,
            optimizer=OPTIMIZER,
            patience=10,
            lr0=0.001,
            device=device,
            workers=0, # workers=0 для MPS
            amp=amp_enabled,
            save_period=1,
            resume=os.path.exists(last_weights)
        )
    except Exception as e:
        print(f"Ошибка при обучении: {e}")
        # Пробуем обучить на CPU
        print("Пробуем обучить на CPU...")
        results = model.train(
            data=data_yaml_path,
            epochs=EPOCHS,
            imgsz=IMGSZ,
            batch=BATCH,
            name=MODEL_NAME,
            optimizer=OPTIMIZER,
            patience=10,
            lr0=0.001,
            device='cpu',
            workers=4,
            amp=False,
            save_period=1
        )

    print("Обучение завершено!")
    return model

def evaluate_model(model, data_yaml_path, device):
    """Оценка модели на тестовых данных"""
    print("\nОценка модели...")

    metrics = model.val(
        data=data_yaml_path,
        split='test',
        device='cpu' # Валидация на CPU для стабильности
    )

    print(f"mAP50-95: {metrics.box.map:.4f}")
    print(f"mAP50: {metrics.box.map50:.4f}")
    return metrics

def visualize_detection(model, image_path, output_dir="detection_results"):
    """Визуализация детекции на изображении"""

```

```

if not os.path.exists(image_path):
    print(f"Изображение не найдено: {image_path}")
    return None, None

print(f"\nВизуализация детекции на {image_path}...")
os.makedirs(output_dir, exist_ok=True)

# Для инференса используем CPU для совместимости
results = model(image_path, device='cpu')

# Сохраняем результат
output_path = os.path.join(output_dir,
f"detected_{os.path.basename(image_path)}")
results[0].save(filename=output_path)

print(f"Результат сохранен в: {output_path}")
return results, output_path

def demonstrate_on_test_images(model):
    """Демонстрация работы модели на тестовых изображениях"""
    print("\nДемонстрация работы детектора на тестовых изображениях...")

    # Используем изображения из тестовой выборки для демонстрации
    test_images_dir = os.path.join(YOLO_DATASET, 'test', 'images')

    if os.path.exists(test_images_dir):
        test_images = [f for f in os.listdir(test_images_dir) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
        if test_images:
            # Выбираем несколько случайных изображений для демонстрации
            demo_images = random.sample(test_images, min(3,
len(test_images)))
            for img_name in demo_images:
                img_path = os.path.join(test_images_dir, img_name)
                visualize_detection(model, img_path)
            else:
                print("В тестовой директории нет изображений")
        else:
            print(f"Тестовая директория не найдена: {test_images_dir}")

    # Также пробуем на пользовательских изображениях
    for test_img in TEST_IMAGES:
        if os.path.exists(test_img):
            visualize_detection(model, test_img)
        else:
            print(f"Тестовое изображение не найдено: {test_img}")

def main():
    """Основная функция"""
    try:
        print("=" * 50)
        print("Лабораторная работа №3. Обучение детекторов объектов")
        print("Версия с поддержкой GPU")
        print("=" * 50)

        # 0. Проверка доступности GPU
        device = check_gpu_availability()

        # 1. Настройка датасета

```

```

        num_classes, class_names, yolo_data_yaml_path =
setup_dataset_from_download()

# 2. Обучение модели
print("\n" + "=" * 50)
print("ЭТАП 2: ОБУЧЕНИЕ МОДЕЛИ")
print("=" * 50)

model = train_yolo_model(num_classes, class_names,
yolo_data_yaml_path, device)

# 3. Оценка модели
print("\n" + "=" * 50)
print("ЭТАП 3: ОЦЕНКА МОДЕЛИ")
print("=" * 50)
metrics = evaluate_model(model, yolo_data_yaml_path, device)

# 4. Демонстрация работы
print("\n" + "=" * 50)
print("ЭТАП 4: ДЕМОНСТРАЦИЯ РАБОТЫ")
print("=" * 50)
demonstrate_on_test_images(model)

print("\n" + "=" * 50)
print("Лабораторная работа завершена!")
print("=" * 50)
print(f"Модель сохранена в: runs/detect/{MODEL_NAME}/")
if hasattr(metrics, 'box'):
    print(f"mAP50-95: {metrics.box.map:.4f}")
    print(f"mAP50: {metrics.box.map50:.4f}")
print(f"Количество классов: {num_classes}")
print(f"Классы: {class_names}")
print(f"Использованное устройство: {device}")

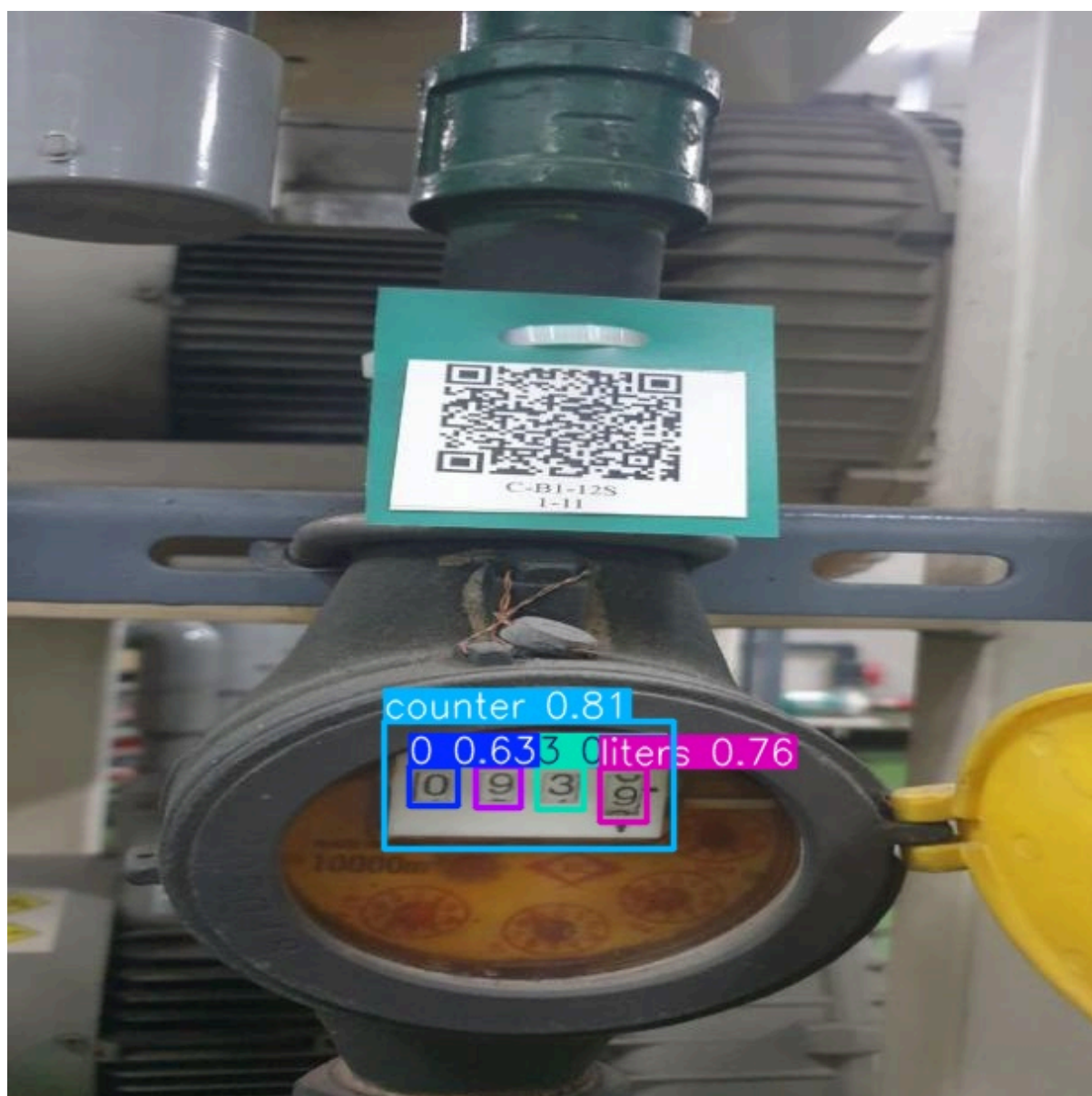
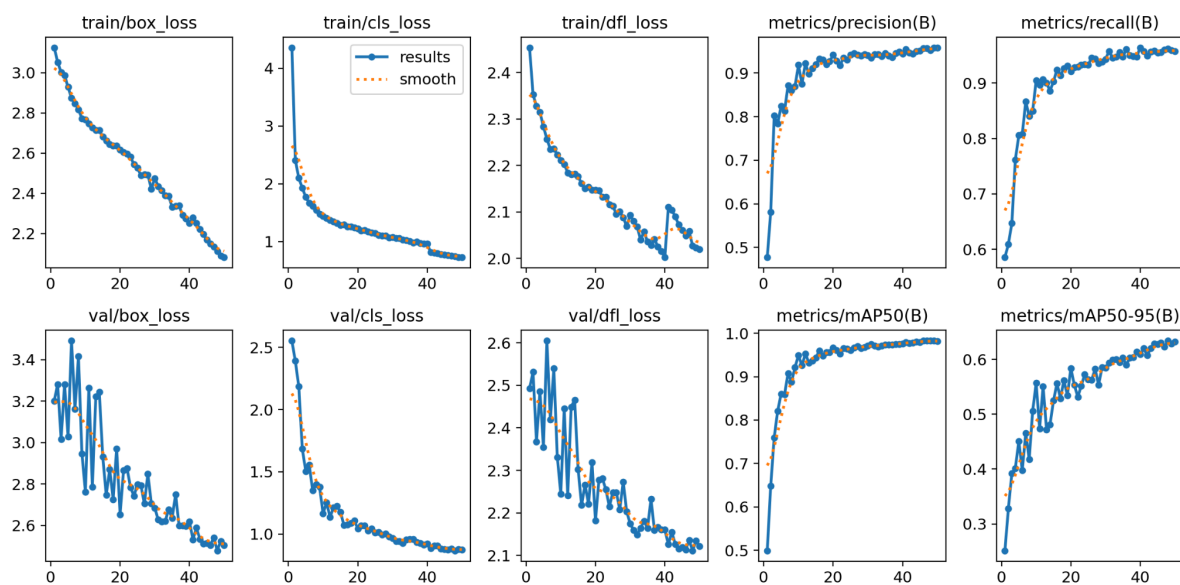
except Exception as e:
    print(f"\nПроизошла ошибка: {e}")
    import traceback
    traceback.print_exc()
    print("\nУбедись, что:")
    print("1. Ты скачал датасет с RoboFlow в формате YOLOv8")
    print("2. Распаковал архив в папку с проектом")
    print("3. Указал правильный путь в переменной
DOWNLOADED_DATASET_PATH")

if __name__ == "__main__":
    main()

```

**Вывод:**





**Вывод:** осуществлять обучение нейросетевого детектора для решения задачи обнаружения заданных объектов