Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Обучение детекторов объектов»

Выполнил:

Студент 4 курса

Группы ИИ-21

Парфеевец И.А.

Проверил:

Крощенко А. А.

Брест 2024

Цель: осуществлять обучение нейросетевого детектора для решения задачи обнаружения дорожных знаков

Ход работы:

Вариант 2

В-т	Детектор
2	YOLOv5n

Код программы:

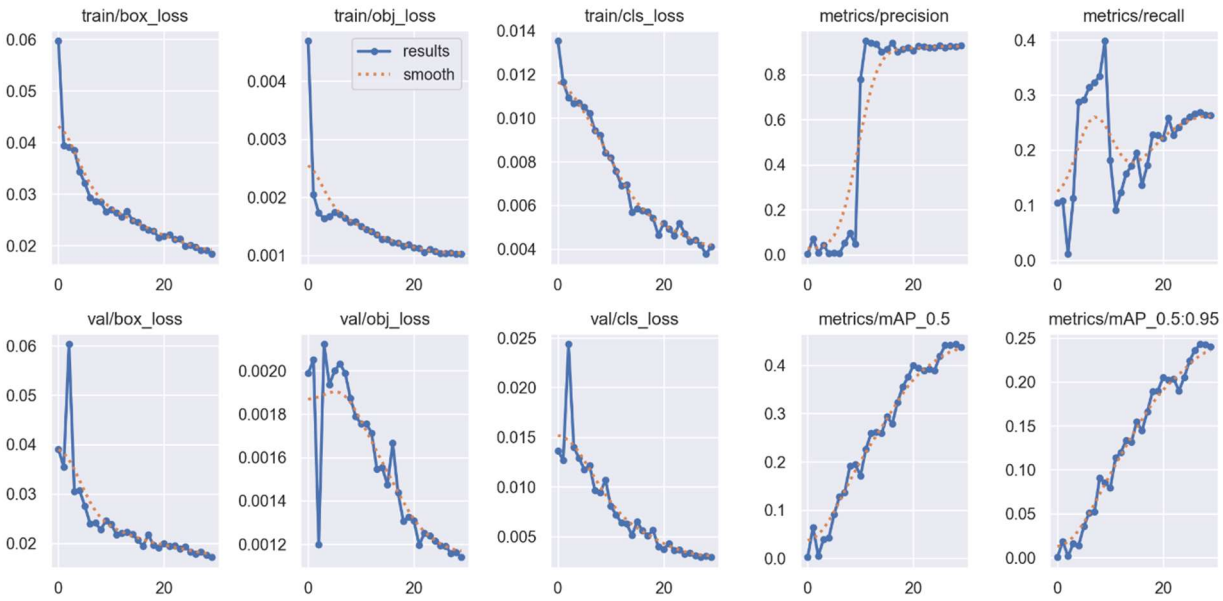
```
import os
import pandas as pd
from pathlib import Path
import shutil
from PIL import Image
import cv2
import torch
from utils.dataloaders import LoadImages
from utils.general import non_max_suppression, scale_boxes
from utils.plots import Annotator
from utils.torch_utils import select_device
from models.common import DetectMultiBackend


# Пути
DATASET_PATH = Path("rtsd-d3-gt")
IMAGES_PATH = Path("rtsd-d3-frames")
OUTPUT_PATH = Path("yolo_data")
YOLO_PATH = Path("yolov5")
MODEL_WEIGHTS = Path("../yolov5_runs/exp9/weights/best.pt")
PREDICTION_INPUTS = ["../Брест день.mp4", "../Брест ночь.mp4"]
PREDICTION_OUTPUTS = ["../output_day.mp4", "../output_night.mp4"]

# Маппинг классов
CLASS_MAPPING = {"blue_border": 0, "blue_rect": 1, "main_road": 2}

# Создание необходимых директорий
OUTPUT_PATH.mkdir(parents=True, exist_ok=True)

def convert_to_yolo(row, img_width, img_height, class_id):
    """
    Конвертирует координаты аннотаций в формат YOLO.
    """
    x_center = (row['x_from'] + row['width'] / 2) / img_width
    y_center = (row['y_from'] + row['height'] / 2) / img_height
    width = row['width'] / img_width
    height = row['height'] / img_height

    if not all(0 <= coord <= 1 for coord in [x_center, y_center, width,
height]):
        return None
```

```
    return f"{class_id} {x_center:.6f} {y_center:.6f} {width:.6f}
{height:.6f}"

def process_annotations():
    """
    Обрабатывает аннотации и готовит данные в формате YOLO.
    """
    for group, class_id in CLASS_MAPPING.items():
        for split in ["train", "test"]:
            csv_file = DATASET_PATH / group / f"{split}_gt.csv"
            if not csv_file.exists():
                print(f"Не найден файл: {csv_file}")
                continue

            df = pd.read_csv(csv_file)
            for _, row in df.iterrows():
                img_path = IMAGES_PATH / split / row['filename']
                if not img_path.exists():
                    continue

                try:
                    with Image.open(img_path) as img:
                        img_width, img_height = img.size
                except Exception as e:
                    print(f"Ошибка при чтении {img_path}: {e}")
                    continue

                yolo_annotation = convert_to_yolo(row, img_width, img_height,
class_id)
                if not yolo_annotation:
                    print(f"Пропущена некорректная аннотация для
{row['filename']}")
                    continue

                output_label = OUTPUT_PATH / split / "labels" /
row['filename'].replace('.jpg', '.txt')
                output_label.parent.mkdir(parents=True, exist_ok=True)
                with open(output_label, 'a') as f:
                    f.write(yolo_annotation + '\n')

                output_image_dir = OUTPUT_PATH / split / "images"
                output_image_dir.mkdir(parents=True, exist_ok=True)
                shutil.copy(img_path, output_image_dir)

def train_yolo():
    """
    Обучает модель YOLOv5.
    """
    os.system(f"""
    python {YOLO_PATH / 'train.py'} --img 1280 --batch 16 --epochs 100 \
    --data {YOLO_PATH / 'data.yaml'} --weights yolov5n.pt --project
yolov5_runs
    """)

def predict_video(source, output):
```

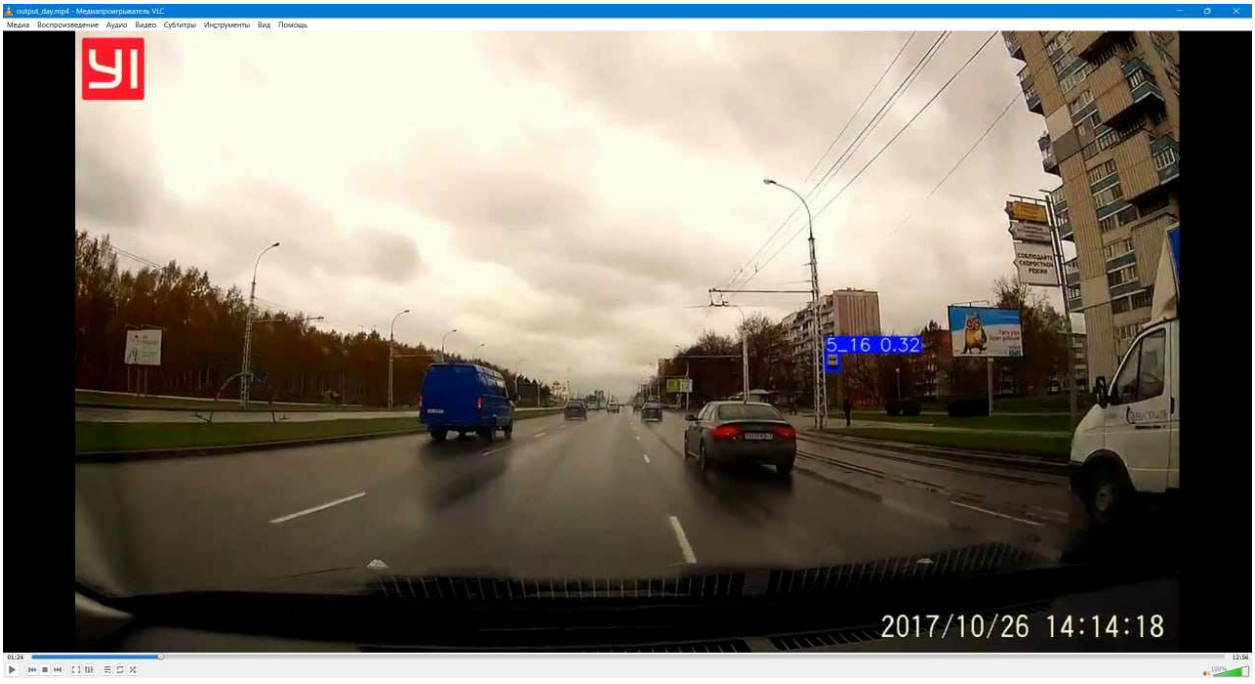
```
    """
    Предсказание объектов на видео и сохранение результата.
    """
    device = select_device('')
    model = DetectMultiBackend(MODEL_WEIGHTS, device=device,
data=str(YOLO_PATH / "data.yaml"))
    stride, names, pt = model.stride, model.names, model.pt
    img_size = 1280

    dataset = LoadImages(source, img_size=img_size, stride=stride, auto=pt)
    vid_writer = None

    for path, img, im0s, vid_cap, _ in dataset:
        img = torch.from_numpy(img).to(device).float() / 255.0
        img = img.unsqueeze(0) if img.ndimension() == 3 else img

        pred = non_max_suppression(model(img))
        for det in pred:
            im0 = im0s.copy()
            annotator = Annotator(im0, line_width=2, example=str(names))
            if det is not None and len(det):
                det[:, :4] = scale_boxes(img.shape[2:], det[:, :4],
im0.shape).round()
                for *xyxy, conf, cls in reversed(det):
                    label = f"{names[int(cls)]} {conf:.2f}"
                    annotator.box_label(xyxy, label, color=(255, 0, 0))
            im0 = annotator.result()

            if vid_writer is None:
                fourcc = cv2.VideoWriter_fourcc(*'mp4v')
                fps = vid_cap.get(cv2.CAP_PROP_FPS) if vid_cap else 30
                w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH)) if vid_cap
else im0.shape[1]
                h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) if vid_cap
else im0.shape[0]
                vid_writer = cv2.VideoWriter(output, fourcc, fps, (w, h))

            vid_writer.write(im0)

    if vid_writer:
        vid_writer.release()

if __name__ == "__main__":
    print("Обработка аннотаций...")
    process_annotations()

    print("Начало обучения YOLO...")
    train_yolo()

    print("Начало предсказания на видео...")
    for src, out in zip(PREDICTION_INPUTS, PREDICTION_OUTPUTS):
        predict_video(src, out)
```

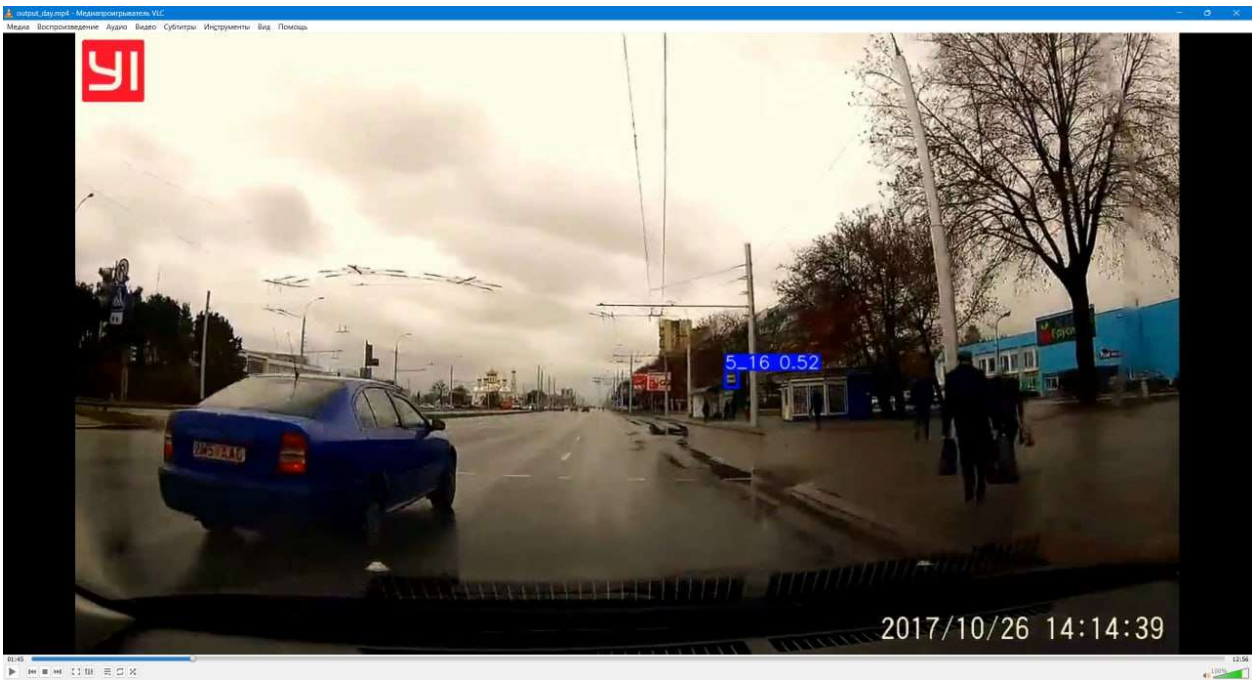
Результат программы:

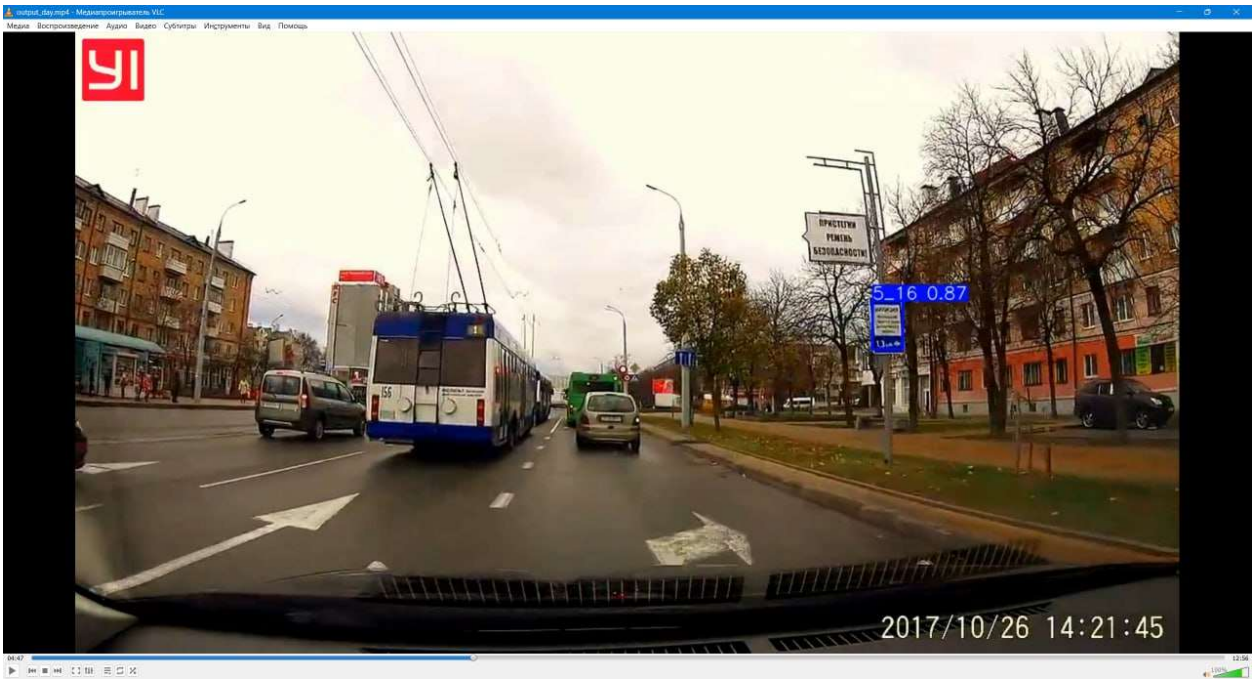
Метрики:



Брест день.mp4:







Ссылка на видео после обработки нейронной сетью - https://disk.yandex.ru/d/aycQrxyx7Uus3w Дата доступа: 27.11.2024.

Вывод: осуществил обучение нейросетевого детектора для решения задачи обнаружения дорожных знаков