`

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

Карагодин Д. Л.

Проверил:

Крощенко А. А.

Брест 2024

`

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

Ход работы:

Вариант 1

В-т	Детектор
1	YOLOv5n

Код программы:

```
import os
import pandas as pd
from pathlib import Path
import shutil
from PIL import Image


DATASET_PATH = "rtsd-d3-gt"
IMAGES_PATH = "rtsd-d3-frames"
OUTPUT_PATH = "yolo_data"


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


os.makedirs(OUTPUT_PATH, exist_ok=True)

def convert_to_yolo(row, img_width, img_height, class_id):
    """
    Конвертирует аннотацию 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 (0 <= x_center <= 1 and 0 <= y_center <= 1 and 0 <= width <= 1 and
0 <= height <= 1):
        return None

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


for group, class_id in CLASS_MAPPING.items():
```

`

```
    print(f"Processing group: {group}")
    for split in ["train", "test"]:
        csv_file = os.path.join(DATASET_PATH, group, f"{split}_gt.csv")
        if not os.path.exists(csv_file):
            print(f"File not found: {csv_file}")
            continue

        df = pd.read_csv(csv_file)

        for _, row in df.iterrows():
            img_path = os.path.join(IMAGES_PATH, split, row['filename'])
            if not os.path.exists(img_path):
                continue


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


            yolo_annotation = convert_to_yolo(row, img_width, img_height,
class_id)
            if yolo_annotation is None:
                print(f"Skipping invalid annotation for image
{row['filename']}")
                continue


            output_file = os.path.join(OUTPUT_PATH, split, "labels",
row['filename'].replace('.jpg', '.txt'))
            os.makedirs(os.path.dirname(output_file), exist_ok=True)
            with open(output_file, 'a') as f:
                f.write(yolo_annotation + '\n')


            output_image_dir = os.path.join(OUTPUT_PATH, split, "images")
            os.makedirs(output_image_dir, exist_ok=True)
            shutil.copy(img_path, output_image_dir)

import os
import torch
print(torch.cuda.is_available())  # Должно вывести True, если CUDA доступна
# Путь к YOLOv5
YOLO_PATH = "yolov5"

# Команда для обучения
train_cmd = f"""
python {YOLO_PATH}/train.py --img 1280 --batch 16 --epochs 100 \
--data {YOLO_PATH}/data.yaml --weights yolov5n.pt --project yolov5_runs
"""
os.system(train_cmd)

import cv2
```

`

```
import torch
from pathlib import Path
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


YOLO_PATH = Path("")
device = select_device('')
weights = "../yolov5_runs/exp9/weights/best.pt"
data = str(YOLO_PATH / "data.yaml")


model = DetectMultiBackend(weights, device=device, data=data)
stride, names, pt = model.stride, model.names, model.pt
img_size = 1280


def predict_video(source, output):
    dataset = LoadImages(source, img_size=img_size, stride=stride, auto=pt)
    vid_writer = None

    for path, img, im0s, vid_cap, s in dataset:

        img = torch.from_numpy(img).to(device).float() / 255.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)


        pred = model(img)
        pred = non_max_suppression(pred)


        for i, det in enumerate(pred):
            im0 = im0s.copy()
            annotator = Annotator(im0, line_width=2, example=str(names))

            if 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()


predict_video("../Брест день.mp4", "../output_day.mp4")
predict_video("../Брест ночь.mp4", "../output_night.mp4")
```

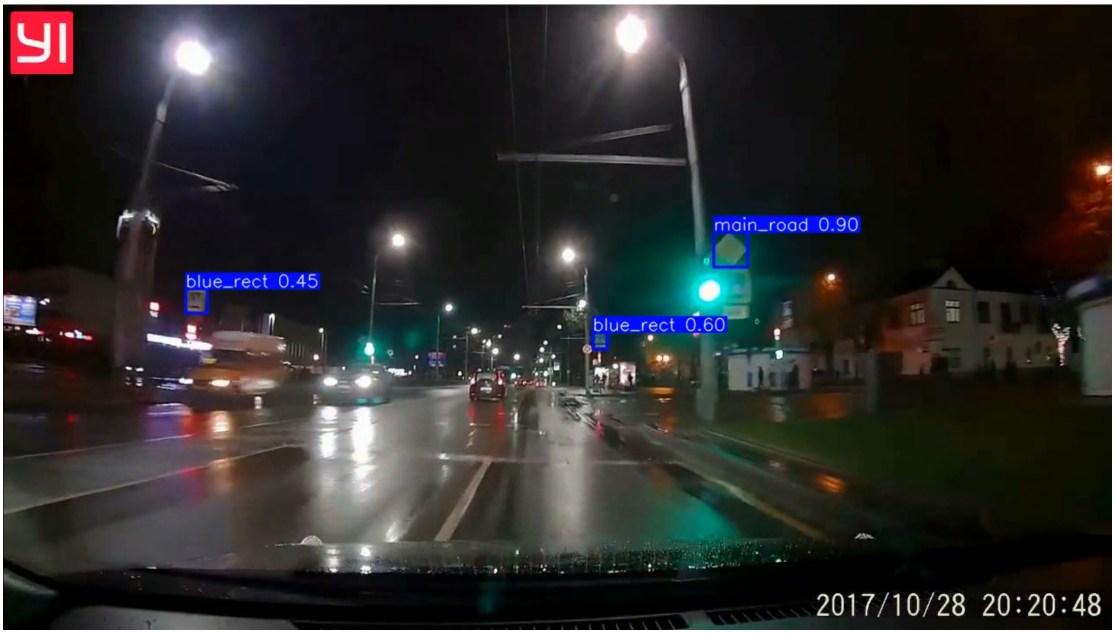
Результат программы:

Брест день.mp4:

`

Брест ночь.mp4

`



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