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()

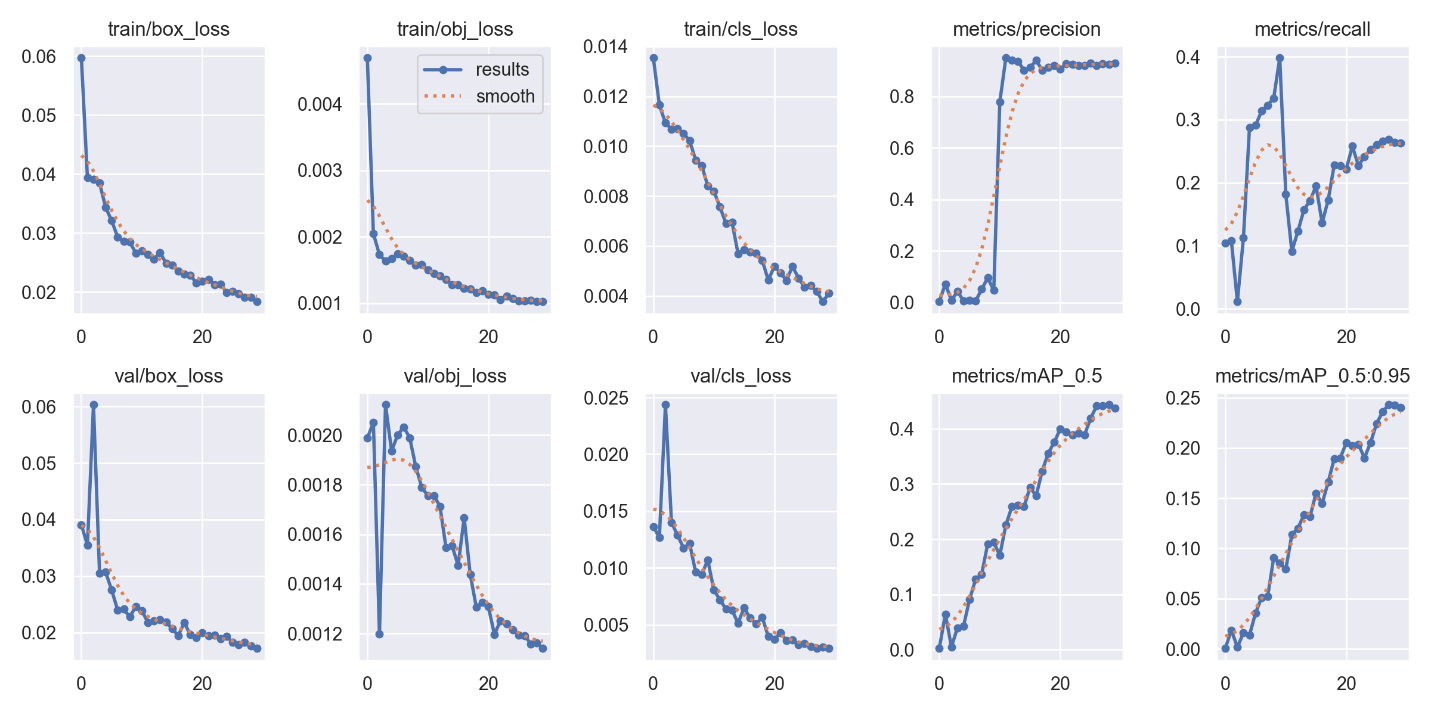
print("Начало предсказания на видео...")

for src, out in zip(PREDICTION\_INPUTS, PREDICTION\_OUTPUTS):

predict\_video(src, out)

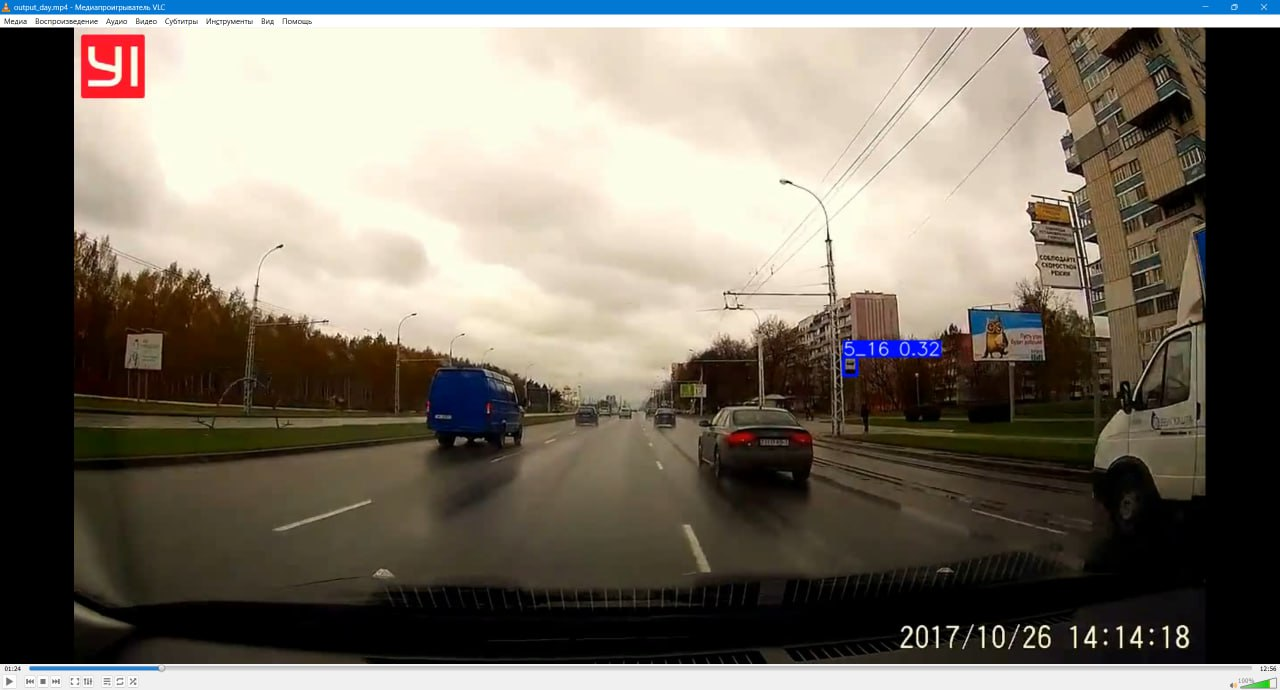
**Результат программы:**

**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