МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Специальность ИИ(з)

Выполнил
А. Ю. Кураш,
студент группы ИИ-24

Проверил
Андренко К.В,
Преподаватель-стажер кафедры ИИТ,
«___k _____2025 г.

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.

8	YOLOv11n	**Номерные знаки авто:** https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11

Выполнение:

Код программы

```
import os
import cv2
import numpy as np
from ultralytics import YOLO
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk


class LicensePlateDetectorApp:
    def __init__(self, master):
        self.master = master
        master.title("YOLOv11n License Plate Detector")

        # Предполагаемый путь к обученной модели YOLOv11
        # Убедитесь, что это путь к вашим лучшим весам (best.pt) после обучения!
        self.model_path =
```

```
"C:\\Users\\User\\OneDrive\\Desktop\\IP_AI_24\\reports\\Kurash\\lab3\\src\\yolov11n.pt"

        # Загрузка модели YOLOv11
        try:
            self.model = YOLO(self.model_path)
            print(f"Модель YOLOv11n загружена из: {self.model_path}")
        except Exception as e:
            print(f"Ошибка при загрузке модели: {e}")
            print("Убедитесь, что путь к модели правильный и она существует.")
            # Если модель не загружена, мы не можем продолжить
            master.destroy()
            return

        self.image_path = None
        self.original_image = None
        self.processed_image = None

        # --- Создание элементов GUI ---

        # Фрейм для кнопок
        self.button_frame = tk.Frame(master)
        self.button_frame.pack(pady=10)

        self.select_button = tk.Button(self.button_frame, text="Выбрать изображение",
command=self.select_image)
        self.select_button.pack(side=tk.LEFT, padx=5)

        self.detect_button = tk.Button(self.button_frame, text="Найти номерной знак",
command=self.detect_license_plate)
        self.detect_button.pack(side=tk.LEFT, padx=5)
        self.detect_button.config(state=tk.DISABLED) # Отключить кнопку до выбора изображения

        # Canvas для отображения изображения
        self.canvas = tk.Canvas(master, bg="lightgray", width=800, height=600)
        self.canvas.pack(pady=10)
```

```
        # Метка для статуса
        self.status_label = tk.Label(master, text="Выберите изображение для начала", bd=1, relief=tk.SUNKEN, anchor=tk.W)
        self.status_label.pack(side=tk.BOTTOM, fill=tk.X)

    def select_image(self):
        self.image_path = filedialog.askopenfilename(
            initialdir=os.path.join(os.path.expanduser("~"), "Desktop"), # Начинаем с рабочего стола
            title="Выберите изображение",
            filetypes=(("Image files", "*.jpg *.jpeg *.png *.bmp"), ("All files", "*.*"))
        )
        if self.image_path:
            self.status_label.config(text=f"Выбрано: {os.path.basename(self.image_path)}")
            self.load_and_display_image(self.image_path)
            self.detect_button.config(state=tk.NORMAL) # Включить кнопку "Detect"
        else:
            self.status_label.config(text="Выбор изображения отменен.")
            self.detect_button.config(state=tk.DISABLED) # Отключить кнопку "Detect"

    def load_and_display_image(self, path):
        # Загружаем изображение с помощью OpenCV
        self.original_image = cv2.imread(path)
        if self.original_image is None:
            self.status_label.config(text=f"Ошибка: Не удалось загрузить изображение {path}")
            return

        self.processed_image = self.original_image.copy() # Копируем для обработки

        # Конвертируем для отображения в Tkinter
        image_rgb = cv2.cvtColor(self.original_image, cv2.COLOR_BGR2RGB)
        image_pil = Image.fromarray(image_rgb)

        # Изменяем размер изображения, чтобы оно поместилось в Canvas
        canvas_width = self.canvas.winfo_width()
```

```
        canvas_height = self.canvas.winfo_height()
        if canvas_width == 1 and canvas_height == 1: # Начальные значения Tkinter
            canvas_width = 800
            canvas_height = 600

        image_pil.thumbnail((canvas_width, canvas_height), Image.LANCZOS)
        self.tk_image = ImageTk.PhotoImage(image_pil)

        self.canvas.delete("all")
        self.canvas.create_image(canvas_width / 2, canvas_height / 2, anchor=tk.CENTER,
image=self.tk_image)
        self.canvas.image = self.tk_image # Сохраняем ссылку

    def detect_license_plate(self):
        if self.original_image is None:
            self.status_label.config(text="Ошибка: Изображение не загружено.")
            return

        self.status_label.config(text="Обнаружение номерных знаков...")
        # Копируем исходное изображение для рисования результатов
        display_image = self.original_image.copy()

        try:
            # Преобразование BGR в RGB для модели (хотя predict часто обрабатывает BGR)
            image_for_inference = cv2.cvtColor(self.original_image, cv2.COLOR_BGR2RGB)
            # Выполняем инференс
            # img_rgb = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
            results = self.model.predict(
                source=image_for_inference,
                conf=0.25, # Минимальный порог достоверности
                imgsz=640, # Размер изображения для инференса (должен соответствовать обучению)
                save=False,
                show=False # Не показывать окно cv2.imshow
            )
            # Отрисовка результатов на изображении
```

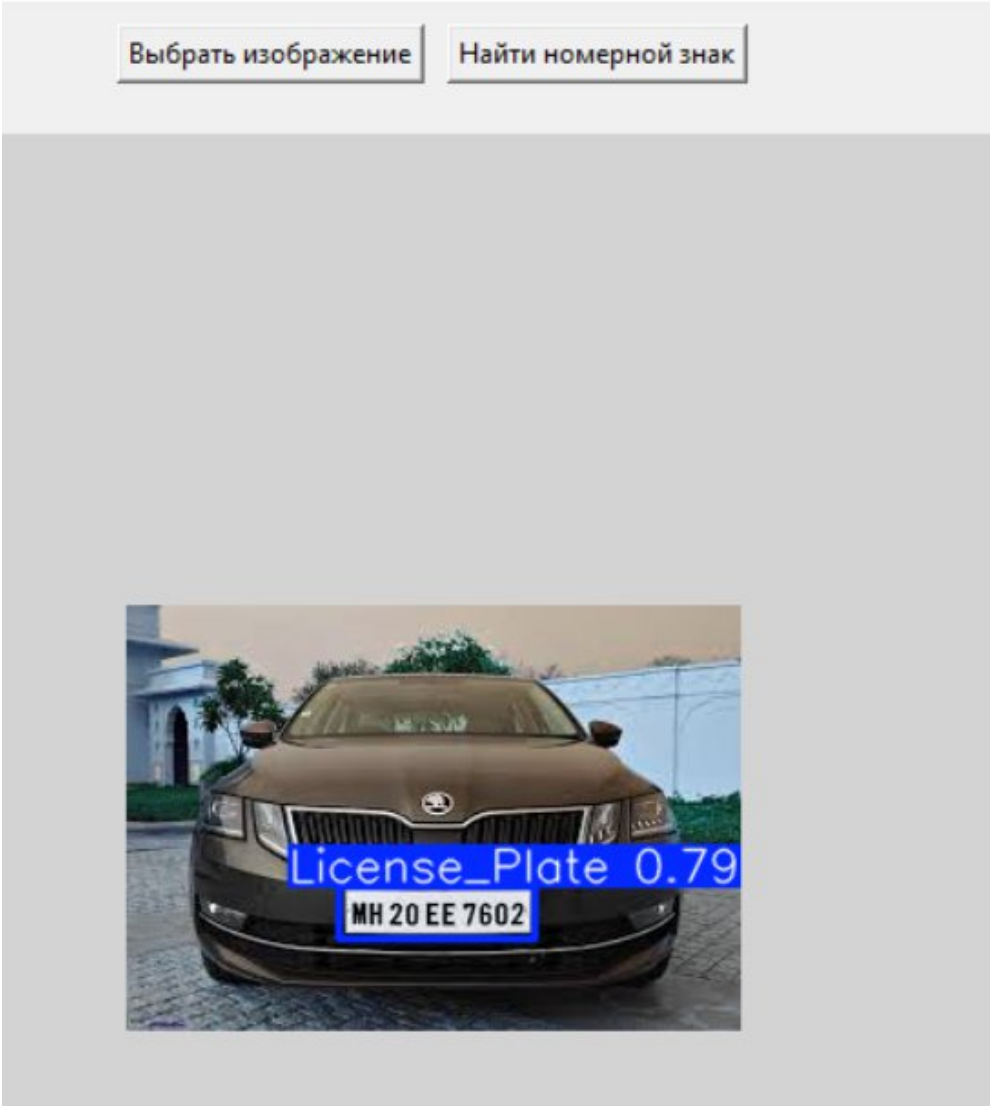
```
            found_plates = False
            for r in results:
                # `r.plot()` возвращает изображение NumPy с отрисованными рамками
                # Это самый простой способ визуализации с Ultralytics
                display_image = r.plot()
                if len(r.boxes) > 0:
                    found_plates = True
            if found_plates:
                self.status_label.config(text="Обнаружены номерные знаки!")
            else:
                self.status_label.config(text="Номерные знаки не обнаружены.")

            # Обновляем отображение на Canvas
            self.update_canvas_with_image(display_image)
        except Exception as e:
            self.status_label.config(text=f"Ошибка при обнаружении: {e}")
            print(f"Ошибка при обнаружении: {e}")

    def update_canvas_with_image(self, img_bgr):
        # Конвертируем изображение OpenCV (BGR) в формат PIL (RGB)
        img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
        image_pil = Image.fromarray(img_rgb)
        # Изменяем размер для отображения
        canvas_width = self.canvas.winfo_width()
        canvas_height = self.canvas.winfo_height()
        if canvas_width == 1 and canvas_height == 1:
            canvas_width = 800
            canvas_height = 600

        image_pil.thumbnail((canvas_width, canvas_height), Image.LANCZOS)
        self.tk_image = ImageTk.PhotoImage(image_pil)
        self.canvas.delete("all")
        self.canvas.create_image(canvas_width / 2, canvas_height / 2, anchor=tk.CENTER,
image=self.tk_image)
        self.canvas.image = self.tk_image
```

```
if __name__ == "__main__":
    root = tk.Tk()
    app = LicensePlateDetectorApp(root)
    root.mainloop()
```



Вывод: Я изучил работу с моделями YOLO и научился настраивать их для определенных задач.