

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Специальность ИИ24

Выполнил
Д. Д. Крупич,
студент группы ИИ24

Проверила
К. В. Андренко,
Преподаватель-стажер кафедры
ИИТ,
«___» _____ 2025 г.

Брест 2025

Цель: осуществлять обучение нейросетевого детектора для решения задачи обнаружения заданных объектов

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.

Вариант 7:

7	YOLOv11n	Люди: https://universe.roboflow.com/leo-ueno/people-detection-o4rdr/dataset/10
---	----------	---

Выполнение:

Код программы:

```
import tkinter as tk
from tkinter import filedialog, Label, Button, Frame
from PIL import Image, ImageTk
from ultralytics import YOLO
import cv2

class DetectorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("YOLOv11: Детектор людей")
        self.root.geometry("1000x600")

        self.model = YOLO("best.pt")

        self.btn_load = Button(root, text="Загрузить изображение",
                                command=self.load_img, font=("Arial", 14), bg="#ddd")
        self.btn_load.pack(pady=20)

        self.image_frame = Frame(root)
        self.image_frame.pack(expand=True, fill="both")

        self.lbl_original = Label(self.image_frame, text="Здесь будет оригинал",
                                   bg="gray", width=50, height=25)
```

```

self.lbl_original.pack(side="left", padx=20, expand=True)

self.lbl_result = Label(self.image_frame, text="Здесь будет результат",
bg="gray", width=50, height=25)
self.lbl_result.pack(side="right", padx=20, expand=True)

def load_img(self):
    path = filedialog.askopenfilename(filetypes=[("Images", "*.jpg *.jpeg *.png")])
    if not path:
        return

    original_pil = Image.open(path)
    original_pil.thumbnail((450, 450))
    self.img_orig_tk = ImageTk.PhotoImage(original_pil)
    self.lbl_original.config(image=self.img_orig_tk, text="", width=0, height=0)

    results = self.model.predict(path, conf=0.5)

    res_array = results[0].plot()
    res_array = cv2.cvtColor(res_array, cv2.COLOR_BGR2RGB)

    result_pil = Image.fromarray(res_array)
    result_pil.thumbnail((450, 450))
    self.img_res_tk = ImageTk.PhotoImage(result_pil)
    self.lbl_result.config(image=self.img_res_tk, text="", width=0, height=0)

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

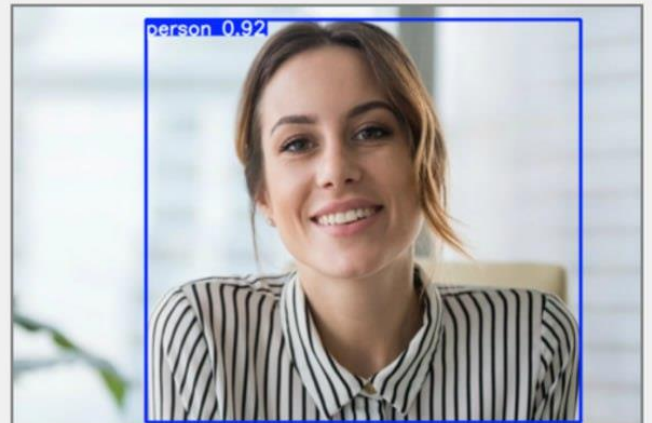
```

Результаты работы программы

Загрузить изображение



Загрузить изображение



Загрузить изображение



Вывод: осуществил обучение нейросетевого детектора для решения задачи обнаружения заданных объектов.