Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «МКиИТ»

Отчёт по производственной практике (ИИ)

Вариант 14: Детекция медведей вблизи населенных пунктов

Выполнил:

Студент группы БФИ2101

Фаттахов Т.Ф.

Руководитель:

Ассистент. каф. МКиИТ

Морочко А.А.

Москва, 2025

**Индивидуальное задание**

Задание: Применение искусственных нейронных сетей для решения задач классификации, детектирования и сегментации в компьютерном зрении с использованием предобученных моделей для детекции медведей вблизи населенных пунктов*.*

1. Цель практики

Освоить полный цикл разработки системы искусственного интеллекта для обработки изображений: от выбора архитектуры нейронной сети до внедрения модели в веб-приложение с визуализацией результатов.

Этапы выполнения:

1. Ознакомление с вариантом (можно взять свой по согласованию с руководителем практики).
2. Подбор архитектуры нейронной сети.
3. Реализация веб-интерфейса. Создать простое веб-приложение. Реализовать: Загрузку изображений/видео/стрим с открытой камеры, кнопку запуска обработки, вывод результатов.
4. Интеграция предобученной модели.
5. Вывод статистики в веб-интерфейсе. Реализовать сохранение истории запросов (в JSON или БД), генерацию отчёта (PDF/Excel).

**Реализация**

Задание предполагает детекцию медведей вблизи населенных пунктов с использованием предобученной модели искусственной нейронной сети.

Для данной задачи была выбрана модель YOLOv5. Для серверной части был выбран язык программирования Python.

Для выполнения поставленной задачи, необходимо предобучить модель путем анализа соответствующих изображений. Для этого был собран DataSet, состоящий из 400 различных изображений медведей. Все изображения были проаннотированы.

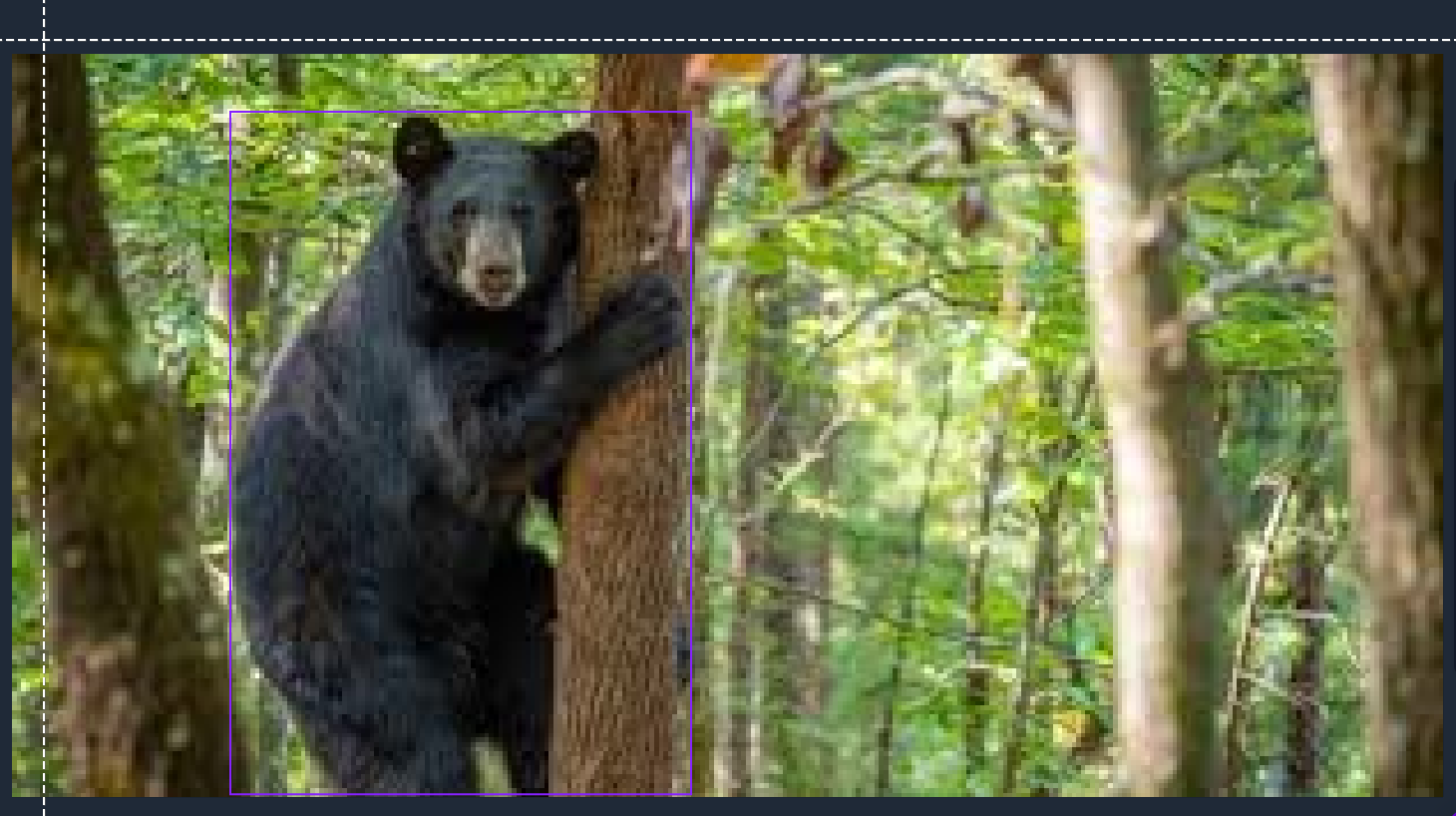


Рисунок 1 – Пример аннотации данных

Затем было произведено обучение модели нейронной сети по заготовленным данным.

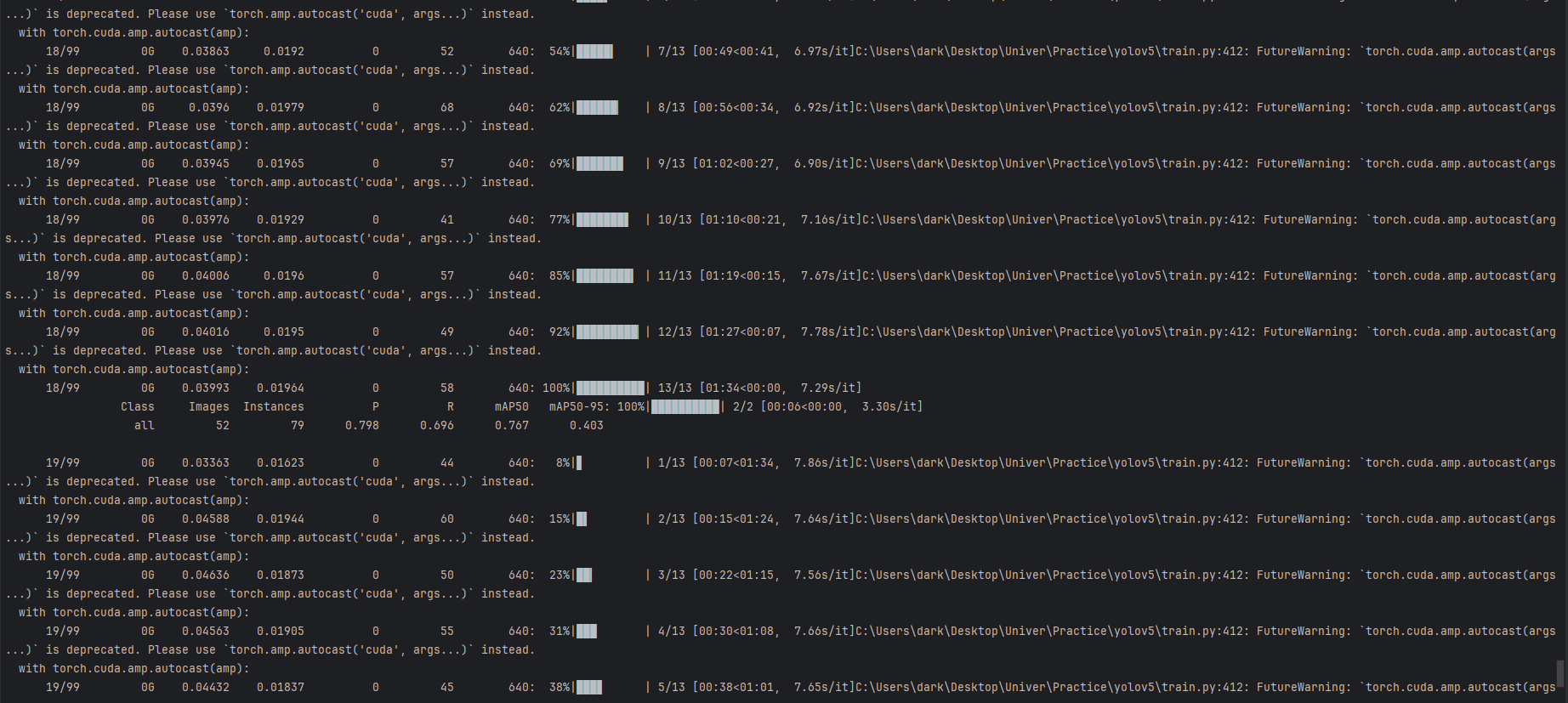


Рисунок 2 – Процесс обучения модели

После обучения модели создаем веб-интерфейс. Для этого был выбрана библиотека Streamlit. Она позволяет создавать веб-приложения для машинного обучения и анализа данных. Для работы с изображениями будем использовать библиотеку PIL. Для совместимости путей между разными ОС используются библиотеки pathlib и platform. Для числовых операций используется и работой с моделью используется NumPy и PyTorch.

Импортируем необходимые инструменты.

import streamlit as st  
from PIL import Image  
import numpy as np  
import torch  
import sys  
import os  
import cv2  
import platform  
import pathlib

Импортируем YoloV5.

yolov5\_path = os.path.join(os.path.dirname(\_\_file\_\_), 'yolov5')  
  
if yolov5\_path not in sys.path:  
 sys.path.append(yolov5\_path)  
  
try:  
 from models.common import DetectMultiBackend  
 from utils.general import non\_max\_suppression, scale\_boxes, xyxy2xywh  
 from utils.torch\_utils import select\_device  
 from utils.augmentations import letterbox  
 from utils.plots import Annotator, colors  
except ImportError as e:  
 st.error(  
 f"Не удалось импортировать модули: {e}")  
 st.stop()

Далее настраиваем конфигурацию для обученной модели.

MODEL\_WEIGHTS\_PATH = 'best.pt'  
IMG\_SIZE = (640, 640)  
CONF\_THRESHOLD = 0.25  
IOU\_THRESHOLD = 0.45

Функция для загрузки обученной модели. Для предотвращения повторной загрузки используем кэширование Streamlit.

@st.cache\_resource  
def load\_yolov5\_model(weights\_path=MODEL\_WEIGHTS\_PATH):  
 try:  
 device = select\_device('') # '' для автоматического выбора (CPU/GPU)  
 model = DetectMultiBackend(weights\_path, device=device, dnn=False, data=None, fp16=False)  
 model.eval() # Переводим модель в режим оценки  
 st.success("Успешно!")  
 return model, device  
 except Exception as e:  
 st.error(f"Ошибка при загрузке модели: {e}")  
 st.stop()

Функция для преобразования изображений в формат, подходящий для модели.

def preprocess\_image(image\_pil, img\_size, stride):  
  
 img\_np = np.array(image\_pil)  
 img\_np\_bgr = cv2.cvtColor(img\_np, cv2.COLOR\_RGB2BGR)  
  
 img = letterbox(img\_np\_bgr, img\_size, stride=stride, auto=True)[0]  
 img = img.transpose((2, 0, 1))[::-1]  
 img = np.ascontiguousarray(img)  
 return img\_np\_bgr, img

Функция предсказания с помощью обученной модели.

def run\_inference(model, device, preprocessed\_img\_tensor):  
  
 preprocessed\_img\_tensor = torch.from\_numpy(preprocessed\_img\_tensor).to(device)  
 preprocessed\_img\_tensor = preprocessed\_img\_tensor.float() # uint8 to fp16/32  
 preprocessed\_img\_tensor /= 255.0 # 0 - 255 to 0.0 - 1.0  
 if preprocessed\_img\_tensor.ndimension() == 3:  
 preprocessed\_img\_tensor = preprocessed\_img\_tensor.unsqueeze(0) # Добавляем размерность пакета  
  
 pred = model(preprocessed\_img\_tensor, augment=False, visualize=False)  
 pred = non\_max\_suppression(pred, conf\_thres=CONF\_THRESHOLD, iou\_thres=IOU\_THRESHOLD, classes=None, agnostic=False,  
 max\_det=1000)  
 return pred, preprocessed\_img\_tensor

Функция отображения итогового результата.

def draw\_detections(original\_img\_bgr, detections, img\_shape\_for\_scaling, names):  
  
 annotator = Annotator(original\_img\_bgr.copy(), line\_width=2, example=str(names))  
 detections\_found = False  
  
 for i, det in enumerate(detections):  
 if len(det):  
 detections\_found = True  
 det[:, :4] = scale\_boxes(img\_shape\_for\_scaling, det[:, :4], original\_img\_bgr.shape).round()  
  
 for c in det[:, 5].unique():  
 n = (det[:, 5] == c).sum()   
 st.write(f"- Обнаружен {int(n)} Медведей")  
  
 for \*xyxy, conf, cls in reversed(det):  
 label = f'{names[int(cls)]}'  
 annotator.box\_label(xyxy, label, color=colors(int(cls), True))  
  
 result\_img = annotator.result()  
 return result\_img, detections\_found

Основная функция запуска веб-интерфейса. Здесь задается стиль интерфейса, устанавливаются заголовки. Загрузка модели.

def main():  
  
 st.set\_page\_config(page\_title="Детектирование медведей вблизи населенных пунктов.", layout="centered")  
  
 st.title("Детектирование медведей вблизи населенных пунктов.")  
 st.write("Загрузите изображение, чтобы обнаружить на нем медведей.")  
  
 model, device = load\_yolov5\_model()  
 stride, names, pt = model.stride, model.names, model.pt   
  
 uploaded\_file = st.file\_uploader("Выберите изображение...", type=["jpg", "jpeg", "png"])  
  
 if uploaded\_file is not None:  
 image = Image.open(uploaded\_file)  
 st.image(image, caption="Загруженное изображение", use\_container\_width=True)  
 st.write("")  
 st.write("Выполняется предсказание...")  
  
 original\_img\_bgr, preprocessed\_img\_tensor = preprocess\_image(image, IMG\_SIZE, stride)  
  
 pred, img\_shape\_for\_scaling = run\_inference(model, device, preprocessed\_img\_tensor)  
  
 result\_img, detections\_found = draw\_detections(original\_img\_bgr, pred, img\_shape\_for\_scaling.shape[2:], names)  
  
 if detections\_found:  
 st.image(result\_img, caption="Результат обнаружения", use\_container\_width=True)  
 else:  
 st.write("Медведи не обнаружены на изображении.")

В результате выполнения представленного кода открывается веб-страница. Пользователь может прикрепить файл для работы с ним. Когда пользователь прикрепил файл, он будет отображен ниже, и пользователь увидит сообщение о успешной загрузке. Затем изображение будет проверено моделью на наличие медведей. В итоге пользователь увидит количество медведей на изображении, а также само изображение с отмеченными объектами.

**Результат работы**

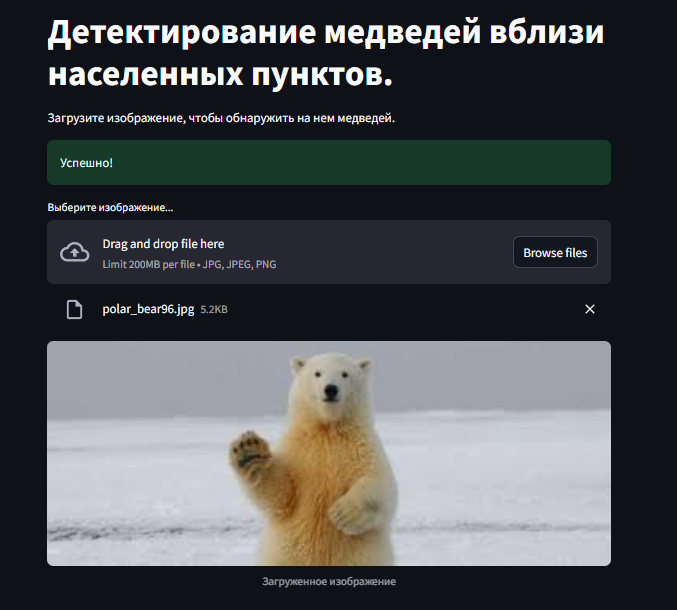


Рисунок 3 – Пример работы веб-интерфейса



Рисунок 4 – Пример отображения результата

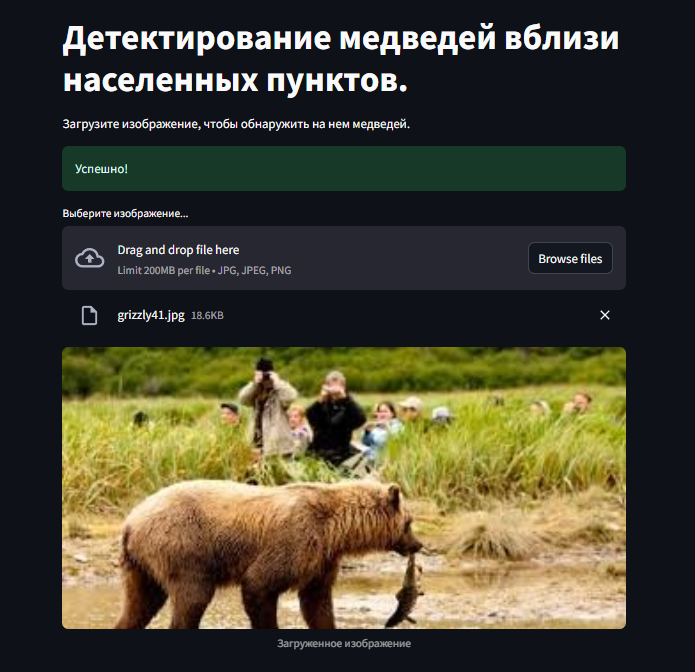


Рисунок 5 – Пример работы веб-интерфейса

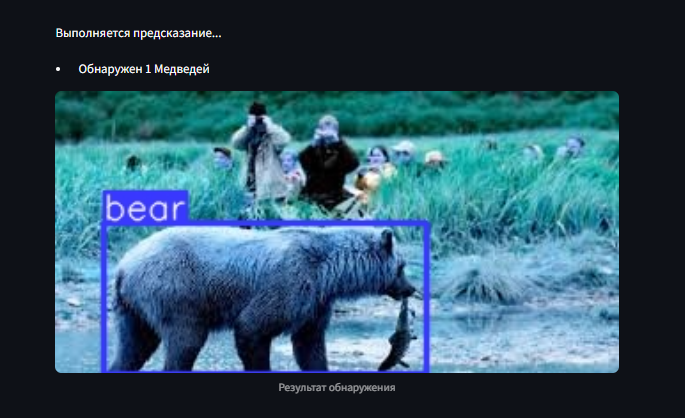


Рисунок 6 – Пример отображения результата

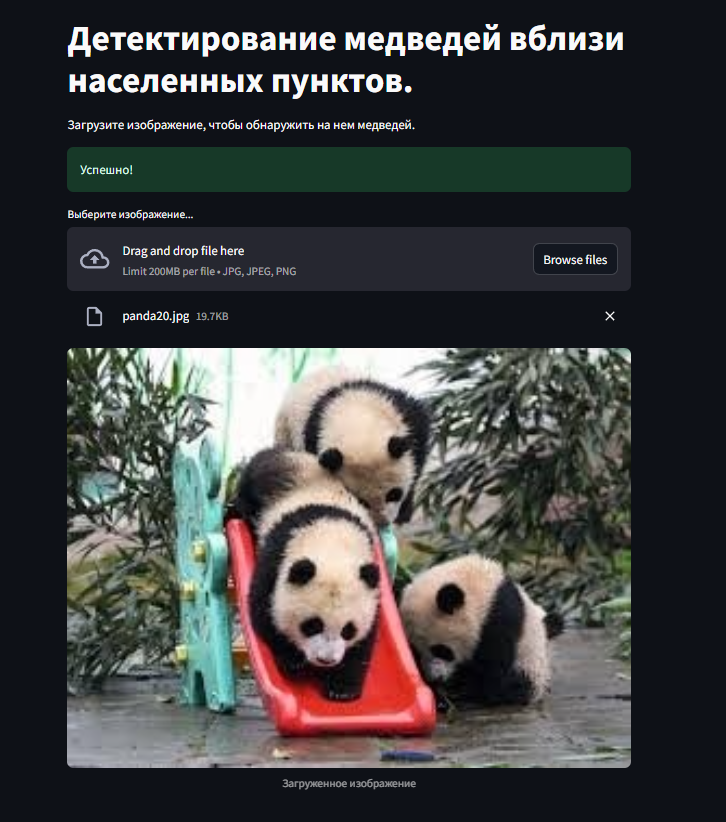


Рисунок 7 – Пример работы веб-интерфейса

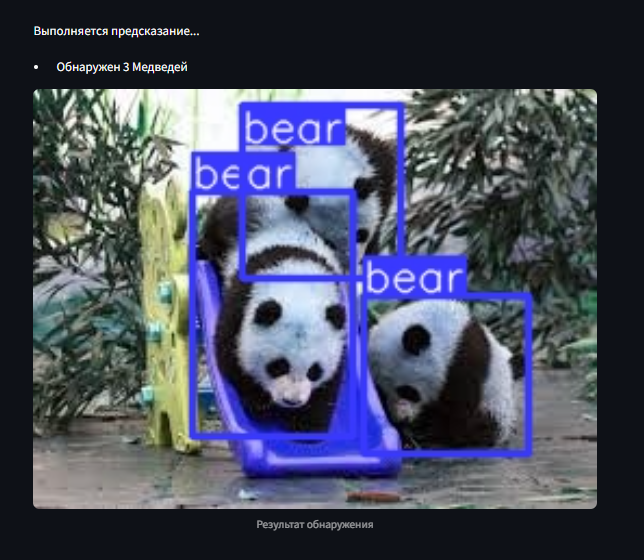


Рисунок 8 – Пример отображения результатов

**Заключение**

В результате выполнения работы освоен полный цикл разработки системы искусственного интеллекта для обработки изображений: от выбора архитектуры нейронной сети до внедрения модели в веб-приложение с визуализацией результатов. Освоено применение искусственных нейронных сетей для решения задач классификации, детектирования и сегментации в компьютерном зрении с использованием предобученных моделей для детекции медведей вблизи населенных пунктов*.*