|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**ПО ДОМАШНЕМУЗАДАНИЮ № 3**

**По дисциплине «Методы поддержки принятия решений»**

**МЕТОДЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВЕ МЕТОДОВ ВЫЧИСЛИТЕЛЬНОГО ИНТЕЛЛЕКТА**

Студент ИУ5-75б  А.Е. Сорокин

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель  Ю.Н. Кротов

(Подпись, дата) (И.О.Фамилия)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва – 2023

**Цель работы:** получить навыки по созданию веб-приложения, работающее с предобученной моделью по классификации изображений.

**Задание:** создать веб-приложение для сбора изображений с использованием предварительно обученной модели на основе датасета cifar100.

**Вариант:** 9

**Классы:** 75, 9, 39

Выполнение работы:

**1) Модель ONNX**

Создадим локальный ноутбук, импортируем необходимые фреймворки. С помощью torch.device определяется устройство (GPU, если доступен, иначе CPU), на котором будет выполняться обработка:

device = torch.device('cuda'  if torch.cuda.is\_available()  else  'cpu')

Загрузим соответствующую модель:

model = torch.hub.load("chenyaofo/pytorch-cifar-models",

    'cifar100\_resnet20',

    pretrained=True)model.to(device)

Используется torch.hub.load для загрузки предобученной модели ResNet20 для CIFAR100.

Модель загружается с предобученными весами (pretrained=True).

CIFAR-100 – это набор данных, состоящий из 100 классов, предназначенных для классификации.

Модель перемещается на определенное устройство (GPU или CPU):

model.to(device)

**Создание тензора для входных данных:**

Создает случайный тензор размером 1x3x32x32 (для изображения CIFAR-100) и перемещает его на устройство.

x = torch.randn(1,  3,  32,  32, requires\_grad=True).to(device)

Загрузим модель на заранее созданный девайс и экспортируем модель, сохранив ее в формате ONNX:

torch.onnx.export(model,  # модель

                    x,  # входной тензор (или кортеж нескольких тензоров)

                    "cifar100\_CNN\_RESNET20.onnx",  # куда сохранить (либо путь к файлу либо fileObject)

                    export\_params=True,  # сохраняет веса обученных параметров внутри файла модели

                    opset\_version=9,  # версия ONNX

                    do\_constant\_folding=True,  # следует ли выполнять укорачивание констант для оптимизации

                    input\_names = ['input'],  # имя входного слоя

                    output\_names = ['output'],  # имя выходного слоя

                    dynamic\_axes={'input'  :  {0  :  'batch\_size'},  # динамичные оси, в данном случае только размер пакета

                    'output'  :  {0  :  'batch\_size'}})

**2) Web-приложение для классификации изображения.**

Инициализируем Django приложение командой в терминале:

django-admin startproject DZ\_Web

Создадим в корне проекта папку media, в которой добавим папки images (для сохранения изображений) и models, в которую сохраним модель ONNX, обученную в первой части.

В settings.py добавим пути до статических файлов (модель для классификации изображений, сами изображения, а также шаблоны, например, index.html).

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [os.path.join(BASE\_DIR, 'static/templates/')],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

MEDIA\_URL = '/media/'

MEDIA\_ROOT = os.path.join(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))), 'media')

Словарь классов изображений:

imageClassList = {9: 'bottle', 39: 'keyboard', 75: 'skunk'}

Функция для отображения страницы с загрузкой изображения:

def scoreImagePage(request):

    return render(request, 'scorepage.html')

Функция predictImage – функция для обработки загруженного изображения.

Функция predictImageData – функция для предсказания класса изображения.

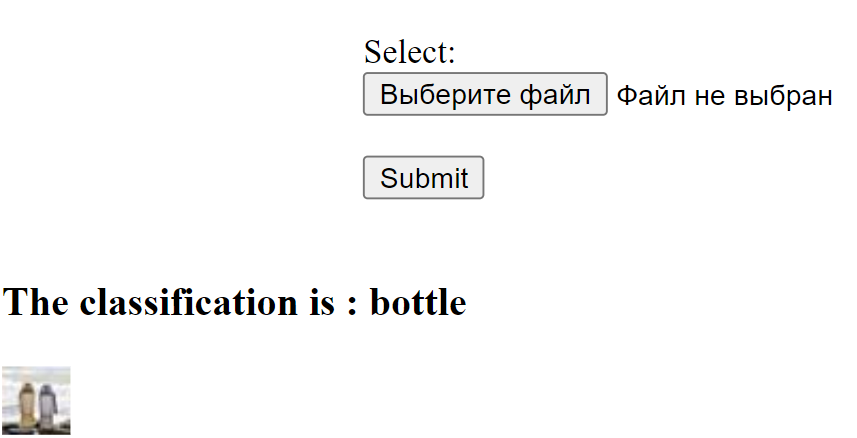
В файле urls.py пропишем маршруты, которые вызывают соответствующие представления.

Добавление маршрута для обработки медиа файлов, таких как изображения.

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**3) Тестирование модели:**

Протестируем прогнозирование модели на классе «Bottle»:



Модель верно классифицирует изображение.

**4) Дополнительное задание:**

Изначально программа не поддерживает кириллическую раскладку в именах файлов изображений и выдает ошибку при загрузке такого файла:



Исправим это, добавив проверку в именах файлов, в случае кириллической раскладки, заменяем имена на случайную строку:

# Загрузка дополнительных библиотек

import uuid

from django.core.exceptions import ValidationError

def validate\_filename(value):

    # Валидация имени файла на наличие кириллических символов

    if any(ord(char) > 127 for char in value):

        raise ValidationError('Имя файла не должно содержать кириллических символов.')

def save\_filename(file):

    try:

        validate\_filename(file.name)

    except ValidationError:

        # Если имя файла содержит кириллические символы, генерируем случайное имя

        random\_filename = f"{uuid.uuid4().hex[:10]}.{file.name.split('.')[-1]}"

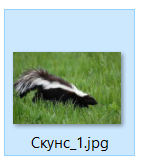
        return random\_filename

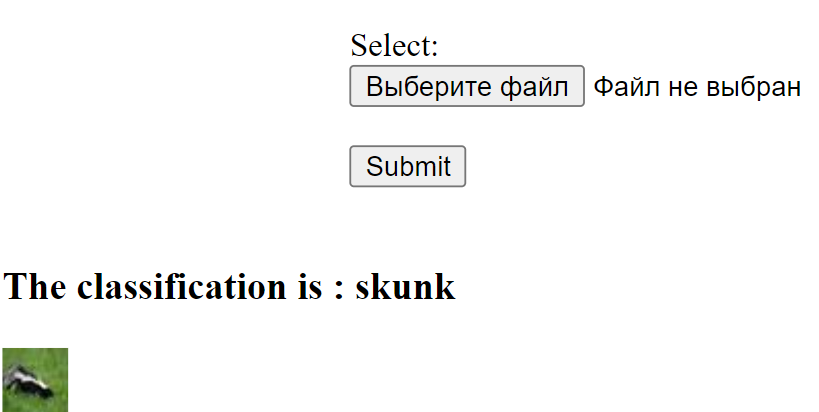
    return file.name

# Вызов функций

filePathName = fs.save('images/'+save\_filename(fileObj), fileObj)

Проверим устойчивость приложения на кириллические имена:





**Вывод:** Были получены навыки по созданию Web-приложения для тестирования предобученной модели классификации изображений.