| 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**

**К ДОМАШНЕМУ ЗАДАНИЮ № 3**

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

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

Студент ИУ5-71б  Д.М. Зелинский

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

Преподаватель  А.Н. Нардид

(Подпись, дата) (И.О.Фамилия)

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

Москва – 2023

**Задание**

Необходимо создать web-приложение для классификации изображений с использованием предобученной модели на основе датасета cifar100. При выполнении задания Вы должны использовать 3-и класса из CIFAR100.

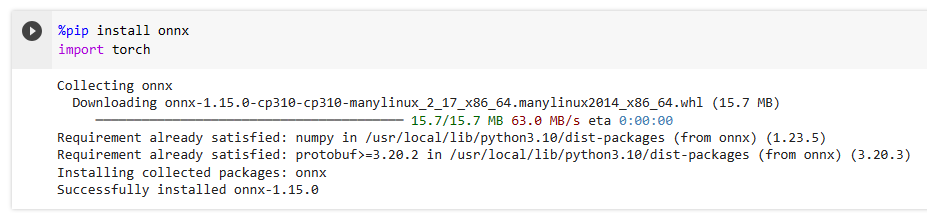
**Исходные данные:**

1. Класс 7: **жук**;
2. Класс 37: **дом**;
3. Класс 67: **морской скат**;
4. Модель: **cifar100\_mobile.**

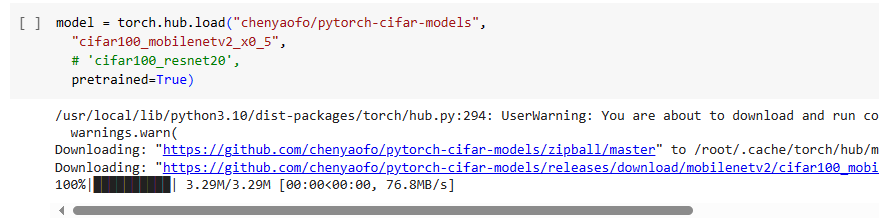
**Выполнение работы**

**Часть 1. Сохранение модели в Google Colab**

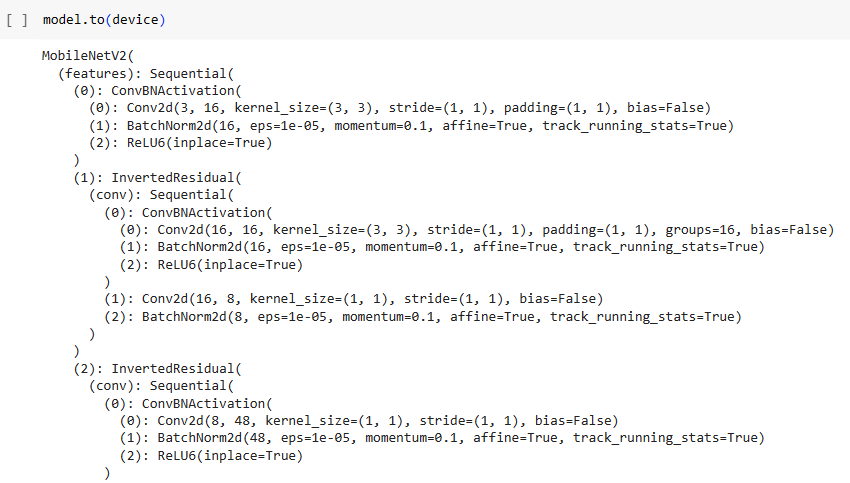
Были импортированы необходимые библиотеки.

Был создан девайс.

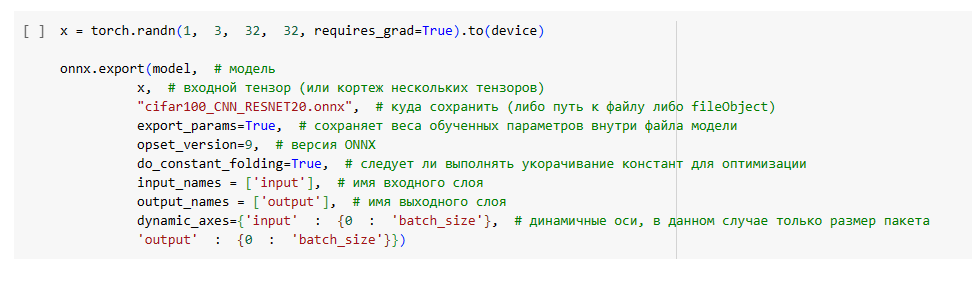
Была загружена модель в соответствии с вариантом.



Модель была загружена на девайс.



Модель была экспортирована и сохранена в формате onnx.



**Часть 2. Web-приложение классификации изображений**

Был создан проект Django в IDE Pycharm. В корне проекта была создана папка “media” для последующего сохранения изображений и файлов формата ONNX. Внутри папки “media” для этого были созданы папки "images" и "models".

Файл settings.py:

*"""  
Django settings for DZ3 project.  
  
Generated by 'django-admin startproject' using Django 4.2.7.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/4.2/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/4.2/ref/settings/  
"""*import os  
  
from pathlib import Path  
  
# Build paths inside the project like this: BASE\_DIR / 'subdir'.  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'django-insecure-)pj+cy^xmy8%fbh1wza!v+-kj6lg&lr6b^imi2k0+9vjo%l\*1('  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'DZ3.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [BASE\_DIR / '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',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'DZ3.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': BASE\_DIR / 'db.sqlite3',  
 }  
}  
  
  
# Password validation  
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/4.2/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/4.2/howto/static-files/  
  
STATIC\_URL = 'static/'  
  
# Default primary key field type  
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'  
  
  
MEDIA\_URL = '/media/'  
MEDIA\_ROOT = os.path.join(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))), 'media')

Файл views.py:

from django.shortcuts import render  
from django.core.files.storage import FileSystemStorage  
import onnxruntime  
import numpy as np  
from PIL import Image  
from io import BytesIO  
import base64  
from torchvision import transforms  
  
imageClassList = {7: 'beetle',  
 37: 'house',  
 67: 'ray', } # Сюда указать классы  
# 7 37 67  
  
def scoreImagePage(request):  
 return render(request, 'scorepage.html')  
  
  
def predictImage(request):  
 fileObj = request.FILES['filePath']  
 fs = FileSystemStorage()  
 filePathName = fs.save('images/' + fileObj.name, fileObj)  
 filePathName = fs.url(filePathName)  
 modelName = request.POST.get('modelName')  
 scorePrediction, img\_uri = predictImageData(modelName, '.' + filePathName)  
 context = {'scorePrediction': scorePrediction, 'filePathName': filePathName, 'img\_uri': img\_uri}  
 return render(request, 'scorepage.html', context)  
  
  
def predictImageData(modelName, filePath):  
 img = Image.open(filePath).convert("RGB")  
 resized\_img = img.resize((32, 32), Image.ANTIALIAS)  
 img\_uri = to\_data\_uri(resized\_img)  
 input\_image = Image.open(filePath)  
 preprocess = transforms.Compose([  
 transforms.Resize(32),  
 transforms.CenterCrop(32),  
 transforms.ToTensor(),  
 transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),  
 ])  
 input\_tensor = preprocess(input\_image)  
 input\_batch = input\_tensor.unsqueeze(0)  
  
  
 sess = onnxruntime.InferenceSession(  
 r'D:\Study\MPPR\DZ3\media\models\cifar100\_CNN\_RESNET20.onnx') # <-Здесь требуется указать свой путь к модели  
 outputOFModel = np.argmax(sess.run(None, {'input': to\_numpy(input\_batch)}))  
 score = imageClassList[outputOFModel]  
  
 return score, img\_uri  
  
  
def to\_numpy(tensor):  
 return tensor.detach().cpu().numpy() if tensor.requires\_grad else tensor.cpu().numpy()  
  
  
def to\_image(numpy\_img):  
 img = Image.fromarray(numpy\_img, 'RG')  
 return img  
  
  
def to\_data\_uri(pil\_img):  
 data = BytesIO()  
 pil\_img.save(data, "JPEG") # pick your format  
  
  
 data64 = base64.b64encode(data.getvalue())  
 return u'data:img/jpeg;base64,' + data64.decode('utf-8')

Файл urls.py:

from django.contrib import admin  
from django.urls import path  
from django.conf.urls.static import static  
from django.conf import settings  
from . import views  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', views.scoreImagePage, name='scoreImagePage'),  
 path('predictImage', views.predictImage, name='predictImage'),  
]  
  
urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Файл scorepage.html:

<!DOCTYPE html>  
<html lang="ru">  
<head>  
<meta charset="windows-1251">  
<title>DZ1</title>  
<style>  
 /\* Add some padding on document's body to prevent the content  
 to go underneath the header and footer \*/ body{  
 padding-top: 60px;  
 padding-bottom: 40px;  
 }  
 .fixed-header, .fixed-footer{  
 width: 100%;  
 position: fixed;  
 background: #333;  
 padding: 10px 0;  
 color: #fff;  
 }  
 .fixed-header{  
 top: 0;  
 }  
 .fixed-footer{  
 bottom: 0;  
 }  
 .container{  
 width: 80%;  
 margin: 0 auto; /\* Center the DIV horizontally \*/  
 }  
 nav a{  
 color: #fff;  
 text-decoration: none;  
 padding: 7px 25px;  
 display: inline-block;  
 }  
</style>  
</head>  
<body>  
 <div class="fixed-header">  
 <div class="container">  
  
 </div> </div> <div class="container">  
 <form action="predictImage" method="post" enctype="multipart/form-data">  
 {% csrf\_token %}  
  
 <div class="col-md-4 col-sm-4">  
 <label for="FilePath">Select:</label>  
 </div> <input name="filePath" type="file"><br><br>  
 <input type="submit" value="Submit" >  
 </form> </div> <div> <br> {% if scorePrediction %}  
 <h3>The classification is : {{scorePrediction}}</h3>  
 {% endif %}  
  
 </div>  
  
 <div> {% if scorePrediction %}  
 <img src="{{ img\_uri }}">  
 {% endif %}  
 </div>  
 <div class="fixed-footer">  
 <div class="container"></div>  
 </div></body>  
</html>

Работа приложения:

|  |  |
| --- | --- |
| Картинка | Результат |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |