МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
к.ф.-м.н., ст. преподаватель кафедры
ММОМ ФГБОУ ВО «ЮУрГГПУ»

_____ А.М. Шарафутдинова

“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2020 г.

Разработка веб-сервиса для геолокации фотографий достопримечательностей с помощью нейронных сетей

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2020.308-061.ВКР

Научный руководитель,
к.ф.-м.н., доцент кафедры СП
_____В. Голодов

Автор работы,
студент группы КЭ-403
_____А.Э. Хлобыстова

Ученый секретарь
(нормоконтролер)
_____И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студенту группы КЭ-403
Хлобыстовой Алине Эдуардовне
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)
Разработка веб-сервиса для геолокации фотографий достопримечательностей с помощью нейронных сетей.
2. **Срок сдачи студентом законченной работы:** 05.06.2020 г.
3. **Исходные данные к работе**
3.1. Noh H. и др. Large-Scale Image Retrieval with Attentive Deep Local Features // Proceedings of the IEEE International Conference on Computer Vision., 2017. С. 3476–3485.
4. **Перечень подлежащих разработке вопросов**
4.1. Провести обзор существующих аналогов и научной литературы.
4.2. Подготовить обучающую, проверочную и тестовую выборку достопримечательностей.
4.3. Разработать архитектуру нейронной сети, занимающейся классификацией достопримечательностей.
4.4. Подготовить наборы данных, состоящих из обучающей и тестовой выборки, для пятнадцати достопримечательностей.
4.5. Разработать архитектуру нейронной сети, определяющей направление съемки.
4.6. Обучить и протестировать спроектированные нейронные сети.
4.7. Разработать и протестировать приложение для определения геолокации фотографии с использованием поиска достопримечательностей.
5. **Дата выдачи задания:** 08.02.2020 г.

Научный руководитель
к.ф.-м.н., доцент кафедры СП В. Голодов

Задание принял к исполнению А.Э. Хлобыстова

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Геотегинг фотографий – это процесс присоединения к изображению географических метаданных, которые содержат информацию о месте, где оно было сделано.

В настоящее время геотегинг довольно популярен среди пользователей социальных сетей и туристических приложений, так как предоставляет широкий спектр информации об их местоположении, позволяет делиться ей с другими, а также дает возможность узнать больше о достопримечательностях и заведениях, находящихся поблизости. Помимо этого, он широко применяется в сферах бизнеса и рекламы. Геотегинг позволяет предпринимателям и компаниям найти покупателей в конкретной области, выяснить какую аудиторию привлекает местоположение и чем аудитория любит заниматься в том или ином месте.

АКТУАЛЬНОСТЬ ПРОБЛЕМЫ

На данный момент существует несколько способов решения задачи определения геолокации изображения: извлечение необходимой информации из EXIF-данных фотографий и использование технологии компьютерного зрения. Первый способ подразумевает наличие GPS-датчика в устройстве, с помощью которого делают фотографию, но так как он присутствует не во всех устройствах, данный подход не является универсальным.

Решая данную задачу вторым способом, можно столкнуться с несколькими проблемами. Во-первых, слишком большой объем исходных данных и схожесть объектов между собой. Во-вторых, внутриклассовая изменчивость, - появление новых объектов, исчезновение или изменение старых. В-третьих, несбалансированность классов. В то время как количество изображений популярных достопримечательностей растет с каждым днем, некоторые малоизвестные места были запечатлены всего несколько раз.

ЦЕЛЬ РАБОТЫ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Целью данного проекта является разработка веб-сервиса для геолокации фотографий достопримечательностей с помощью нейросетевого подхода, который также включает функцию определения направления съемки.

Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор существующих аналогов и научной литературы;

2) подготовить обучающую, проверочную и тестовую выборки достопримечательностей;

3) разработать архитектуру нейронной сети, занимающейся классификацией достопримечательностей;

4) подготовить наборы данных, состоящих из обучающей и тестовой выборки, для пятнадцати достопримечательностей;

5) разработать архитектуру нейронной сети, позволяющей определить направление съемки;

6) обучить и протестировать спроектированные нейронные сети;

7) разработать и протестировать приложение для определения геолокации фотографии с использованием поиска достопримечательностей.

СТРУКТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из введения, пяти разделов, заключения, библиографического списка и трех приложений. Объем работы составляет 49 страниц, объем библиографии – 16 источников, объем приложения – 11 страниц.

В первой главе производится анализ предметной области и обзор аналогичных проектов и существующих решений для реализации проекта, а также обзор существующих решений для создания нейронных сетей.

Во второй главе описаны особенности применения нейронных сетей к задаче обработки изображений и перечислены основные компоненты сверточных нейронных сетей.

В третьей главе определены функциональные и нефункциональные требования к программной системе и представлены варианты использования системы, спроектированы архитектуры сверточных нейронных сетей и графический интерфейс приложения.

Четвертая глава содержит детали реализации веб-сервиса для геолокации фотографий достопримечательностей с помощью нейронных сетей.

В пятой главе приведены результаты тестирования разработанный сверточных нейронных сетей и функционального тестирования разработанного веб-сервиса.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Распознавание достопримечательностей

Главной сложностью распознавания достопримечательностей является то, что в каждом классе существует множество непохожих изображений. В основном это связано с различными ракурсами, и даже наличием вида изнутри (рисунок 1). В то же время иногда встречаются и взаимоисключающие изображения, например, фотографии достопримечательности до и после пожара, сделанные с одного и того же ракурса (рисунок 2).

Рис. 1. Фотографии одной достопримечательности с разных ракурсов

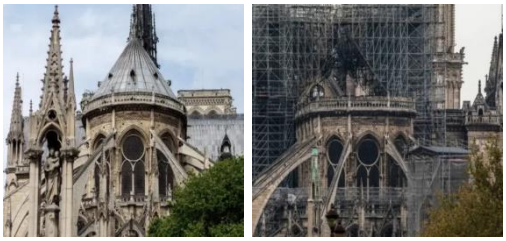

Рис. 2. Фотографии одной достопримечательности с одного ракурса, сделанные в разное время

Помимо этого, достопримечательности крайне разнообразны, ими могут быть исторические или культурные здания, памятники, различные сооружения и природные объекты (рисунок 3).

Рис. 3. Фотографии разных типов достопримечательностей

1.2. Анализ аналогичных проектов и существующих решений для реализации проекта

На сегодняшний день существует несколько приложений и систем, решающих задачу определения местоположения пользователя по фотографии. Ниже перечислены некоторые из них.

Landmark Recognition [1]

Компанией Blippar был выпущен API для распознавания достопримечательностей Landmark Recognition. Интерфейс основан на алгоритмах компьютерного зрения и не пользуется информацией от GPS, благодаря этому приложение может распознавать места по фотографиям и даже рисункам. Основой данного интерфейса является патентованная технология глубокого обучения с исключением ложных срабатываний.

«По данным Blippar, Landmark Recognition имеет вероятность возникновения ошибки ложной тревоги, когда устройство срабатывает без объекта в поле зрения, 0,36% (на основе эксперимента по распознаванию 3000 изображений зданий и мостов), а точность распознавания составляет 91,6%. Оба показателя в компании называют не имеющими аналогов в отрасли. В настоящее время технология «знает» 2164 объекта и развивается» [2].

PlaNet [3]

Некоторое время назад под управлением Тобиаса Веянда, специалиста Google по компьютерному зрению, велась работа над новой системой, способной производить геолокацию изображения с применением глубокого обучения. В отличие от наиболее распространенного способа определения местоположения по достопримечательностям, разработчики данного проекта рассматривают эту задачу с точки зрения классификации: они разделили поверхность Земли на множество ячеек и занялись обучением глубокой нейронной сети, используя миллионы изображений с прикрепленными геотегами.

Благодаря наличию огромного числа публичных фотографий с геоте-

гами и способности PlaNet обучаться на их базе, данная система может предсказать местоположение не только известных достопримечательностей, но и различных природных пейзажей, таких как горы, водопады или пляжи (рисунок 4).

Получив фотографию для обработки, система распределяет по карте Земли вероятностные значения, то есть присваивает каждой ячейке вероятность того, что фотография была сделана в данной области. Красным цветом помечаются области, где предположительно находится искомый объект.

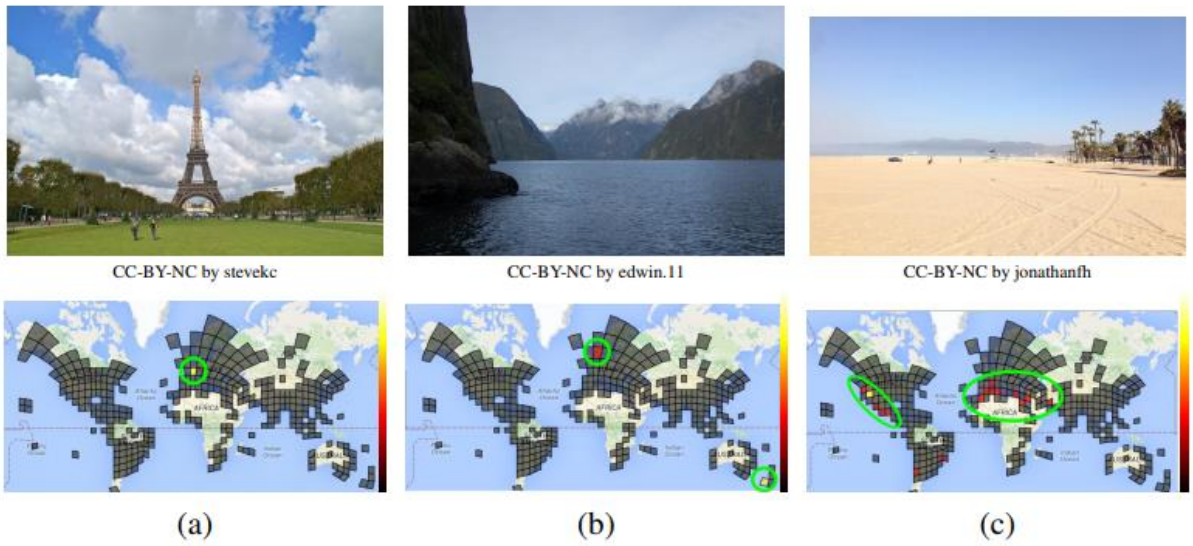

Рис. 4. Пример работы PlaNet

Pic2Map [4]

Данный веб-сервис в отличие от предыдущих аналогов не использует технологию компьютерного зрения, однако может определить местоположение на основе EXIF-данных, являющихся разновидностью метаданных, хранящихся внутри файлов. При условии, что в камере или мобильном устройстве есть GPS-датчик и включена геолокация, географические координаты устройства будут зафиксированы. Pic2Map, используя эти данные, может с предельной точностью установить место съемки и отметить его на карте.

Несмотря на высокую точность определения местоположения использование данного сервиса не всегда возможно, так как многие пользователи мобильных устройств отключают геолокацию из соображений приватности или уменьшения нагрузок на аккумулятор. Так как использование данного решения не универсально, в дальнейшем рассматривается нейросетевой подход.

1.3. Обзор готовых решений для создания нейронных сетей

Самым удобным инструментом для реализации нейронных сетей является язык программирования Python, зарекомендовавший себя в сфере машинного обучения. На нем написано множество библиотек, созданных для работы с нейронными сетями и значительно упрощающих задачу их разработки. Далее представлены некоторые из них:

TensorFlow [5]

Открытая программная библиотека для машинного обучения, разработанная компанией Google и являющаяся одной из самых популярных в своем классе. Она позволяет создавать и обучать различные нейронные сети при помощи понятных высокоуровневых интерфейсов, таких как Keras.

Theano [6]

Theano, одна из первых библиотек для глубокого машинного обучения, разрабатывается с 2007 года группой MILA из Университета Монреаля. Позволяет проводить вычисления как на CPU, так и на GPU. Это отличная библиотека с точки зрения производительности, однако она менее понятна для новичков из-за синтаксиса, поэтому теряет свои преимущества перед фреймворками, с которыми проще работать.

Keras [7]

Еще один популярный инструмент для разработки нейронных сетей. Он позволяет использовать в качестве вычислительного бэкенда другие фреймворки, например, TensorFlow и Theano. По сравнению с другими библиотеками Keras гораздо понятнее, но работает медленнее.

По итогам обзора библиотек было решено использовать Keras, как наиболее понятную и простую для новичков.

Выводы по первому разделу

После изучения существующих проектов и решений этой задачи было принято решение реализовывать определение геолокации фотографии с помощью поиска по достопримечательностям основанном на нейросетевом подходе. Во-первых, в отличие от сервисов и приложений, использующих метаданные фотографий, он не требует наличия и использования GPS-датчика, следовательно, является более универсальным. Во-вторых, требует меньшей базы изображений для обучения нейронной сети, предсказывающей местоположения в любой точке мира, чем система PlaNet.

Также в отличие от рассмотренных аналогов, использующих технологии компьютерного зрения, разрабатываемый веб-сервис будет не только определять местоположение достопримечательности, но и конкретизировать геолокацию фотографии посредством определения направления съемки. Таким образом, область, в которой была сделана фотография, будет значительно сужена.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. Особенности применения нейронных сетей к задаче обработки изображений

Часто задачи распознавания изображений решаются с помощью нейронных сетей. Обычно используются классические нейросетевые архитектуры (многослойный персептрон, сети с радиально-базисной функцией и др.), но в данной задаче их применение неэффективно. Причины низкой эффективности этих подходов уже не раз исследовались, ниже перечислены основные из них.

1. Из-за большой размерности изображений, возрастает размер нейронной сети.

2. Большое количество параметров требует большой тренировочной выборки, а также увеличивает время и вычислительную сложность процесса обучения. Аналогичный эффект вызывает увеличение количества нейронных сетей, использующееся для повышения эффективности работы системы.

3. Отсутствует инвариантность к изменениям масштаба изображения, ракурсов и других геометрических искажений входного сигнала.

Из-за перечисленных недостатков классических нейросетевых структур, в подобных задачах чаще используется метод сверточных нейронных сетей, так как они обеспечивают частичную устойчивость к изменениям масштаба, смещениям и различным ракурсам.

2.2. Основные компоненты сверточных нейронных сетей

Для организации сверточной нейронной сети применяется 3 основных слоя:

1) свертки;

2) пулинга (иначе подвыборки или субдискретизации);

3) полносвязный слой.

Также может использоваться исключающий слой, который «отклю-

чает» нейроны с некоторой вероятностью. Он используется для предотвращения переобучения и ускорения обучения.

Организацию сверточной нейронной сети можно рассмотреть на примере классификации системы распознавания изображений CIFAR-10 со следующей архитектурой [INPUT – CONV – RELU – POOL – FC][6].

1. В INPUT (входные данные) [32×32×3] содержится исходная информация об изображении (в данном случае 32 – ширина, 32 – высота, 3 – количество цветовых каналов).

2. Слой CONV (слой свертки) умножает значения фильтра свертки на исходные значения пикселей изображения, после чего все произведения суммируются. Каждая уникальная позиция введенного изображения производит число. Например, если используется 12 фильтров, то объем будет равен 32*32*12 [32×32×12].

3. Слой RELU (rectified linear unit) применяет поэлементную функцию активации, которая имеет формулу f (x) = max(0, x), устанавливая нулевой порог. То есть в случае, если x < 0, отсекаются ненужные детали в канале путем замены на 0.

4. Слой POOL (слой пулинга) выполняет операцию по понижению частоты дискретизации ширины и высоты (то есть удаление пикселей), в результате чего объем может сократиться до [16×16×12]. Если в ходе предыдущей операции уже были выявлены некоторые признаки, то для дальнейшей обработки достаточно менее подробной картинки, и выполняется нелинейное уплотнение карты признаков.

5. Слой FC (полносвязный слой) выводит N-мерный вектор для определения нужного класса, где N- количество классов.

Ниже для лучшего понимания графически представлена архитектура сверточной нейронной сети LeNet, созданной для распознавания букв (рисунок 5).

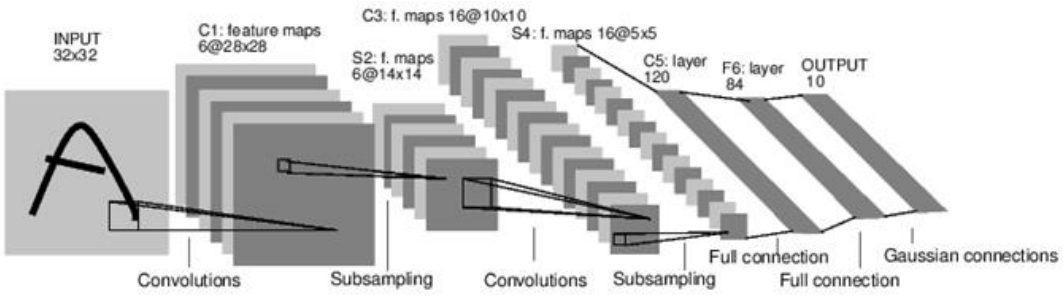

Рис. 5. Архитектура сверточной нейронной сети LeNet

Выводы по второму разделу

Была рассмотрена теоретическая часть о сверточных нейронных сетях: алгоритм работы, информация о трех основных слоях. Также была рассмотрена организация сверточной нейронной сети на примере системы распознавания изображений CIFAR-10.

3. ПРОЕКТИРОВАНИЕ

3.1. Определение требований

Система определения геолокации фотографии будет представлять собой веб-сервис, используя который пользователь сможет загружать изображения и получать информацию об их геолокации и направлении съемки. В ходе проектирования приложения были определены следующие функциональные и нефункциональные требования к разрабатываемому приложению.

Функциональные требования

1. Система должна предоставлять пользователю возможность выбрать и загрузить изображение в формате jpg.

2. Система должна выводить выбранное изображение на экран.

3. Система должна распознавать достопримечательность на изображении и выводить информацию о геолокации изображения.

4. Система должна определять направление съемки и выводить соответствующую информацию.

5. В системе должна присутствовать интерактивная карта, на которой маркером помечается местоположение достопримечательности.

Нефункциональные требования

1. Система должна быть реализована с помощью фреймворка Flask на языке программирования Python.

2. Система должна использовать предварительно обученную нейронную сеть для классификации ста достопримечательностей.

3. Система должна использовать предварительно обученную на нескольких наборах данных нейронную сеть для определения направления съемки.

4. Для работы с нейронными сетями система должна использовать фреймворк Keras.

3.2. Архитектура нейронной сети для классификации достопримечательностей

Для решения задачи классификации достопримечательностей была спроектирована нейронная сеть, архитектура которой представлена в таблице 1.

Табл. 1. Конфигурация сверточной нейронной сети для классификации достопримечательностей

№	Тип слоя	Описание
0	Входной	Изображение размером 128*128 в градациях серого (с одним цветовым каналом)
1	Сверточный	32 карты признаков с ядром свертки 4x4
2	Сверточный	32 карты признаков с ядром свертки 4x4
3	Подвыборки	Размер области 2x2
4	Исключающий	Вероятность исключения 0.5
5	Сверточный	64 карты признаков с ядром свертки 4x4
6	Сверточный	64 карты признаков с ядром свертки 4x4
7	Подвыборки	Размер области 2x2
8	Исключающий	Вероятность исключения 0.5
9	Полносвязный	512 нейронов
10	Исключающий	Вероятность исключения 0.5
11	Полносвязный	100 нейронов

В качестве активационной функции для первого, второго, пятого, шестого и девятого слоев используется полулинейная функция relu. Для последнего полносвязного слоя – функция softmax, позволяющая трактовать выходные значения нейронов как вероятность принадлежности объекта одному из классов.

3.3. Архитектура нейронной сети для классификации направления съемки

Для решения задачи классификации направлений съемки была спроектирована нейронная сеть, архитектура которой представлена в таблице 2.

Табл. 2. Конфигурация сверточной нейронной сети для классификации направлений съемки

№	Тип слоя	Описание
0	Входной	Изображение размером 256*256 в градациях серого (с одним цветовым каналом)
1	Сверточный	64 карты признаков с ядром свертки 4x4
2	Сверточный	64 карты признаков с ядром свертки 4x4
3	Подвыборки	Размер области 2x2
4	Исключающий	Вероятность исключения 0.5
5	Сверточный	128 карт признаков с ядром свертки 4x4
6	Сверточный	128 карт признаков с ядром свертки 4x4
7	Подвыборки	Размер области 2x2
8	Исключающий	Вероятность исключения 0.5
9	Полносвязный	512 нейронов
10	Полносвязный	128 нейронов
11	Полносвязный	32 нейрона
12	Исключающий	Вероятность исключения 0.5
13	Полносвязный	4 нейрона

В качестве активационной функции для первого, второго, пятого, шестого, девятого, десятого и одиннадцатого слоев используется полулинейная функция relu. Для последнего полносвязного слоя – функция softmax, позволяющая трактовать выходные значения нейронов как вероятность принадлежности объекта одному из классов.

3.4. Варианты использования

Для проектирования приложения был использован язык графического описания для объектно-ориентированного программирования UML. Была построена модель взаимодействия внешнего актера с программной системой в виде диаграммы вариантов использования (рисунок 6).

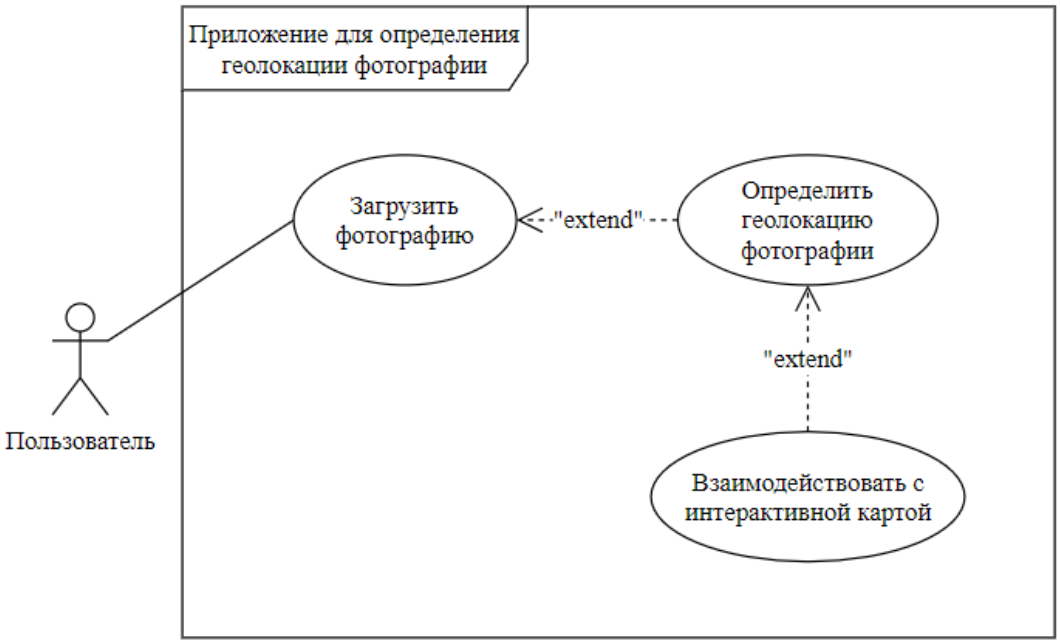

Рис. 6. Диаграмма вариантов использования

С программной системой взаимодействует только один актер – пользователь, использующий программную систему, ему доступны перечисленные ниже варианты использования.

1. Вариант использования «Загрузить фотографию». Пользователь может выбрать и загрузить изображение.

2. Вариант использования «Определить геолокацию фотографии» – пользователь может нажать на кнопку «Определить», после чего запускается процесс классификации выбранного изображения.

3. Вариант использования «Взаимодействовать с интерактивной картой». После определения класса достопримечательности, отображается интерактивная карта, центрированная на ее местоположении, которое помечается маркером. Пользователь может выбрать режим отображения карты

(«Карта» или «Спутник»), перемещаться по карте, менять масштаб, использовать полноэкранный режим, а также режим просмотра улиц.

Спецификация вариантов использования приведена в приложении А.

3.5. Проектирование графического интерфейса пользователя

Веб-приложение должно иметь простой, интуитивно понятный пользовательский интерфейс, обеспечивающий необходимую функциональность.

Интерфейс разрабатываемого приложения включает в себя три основных секции: шапка страницы, основная часть и подвал страницы. Основная часть содержит следующие элементы:

1) кнопка для прикрепления изображения;

2) кнопка определения геолокации, запускающая процесс определения геолокации изображения;

3) поле для вывода информации о геолокации;

4) выбранное изображение;

5) интерактивная карта.

Макет графического интерфейса разрабатываемого веб-приложения представлен на рисунке 7.

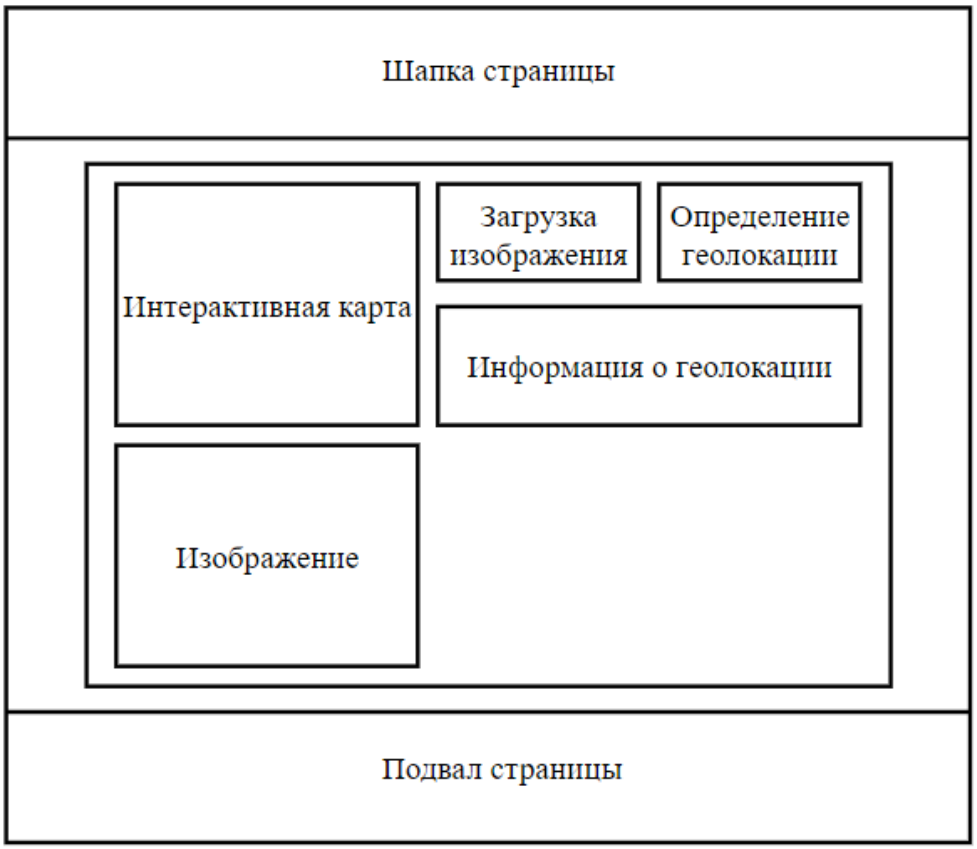

Рис. 7. Макет графического интерфейса

Выводы по третьему разделу

В данном разделе были определены функциональные и нефункциональные требования к системе определения геолокации фотографии. На основе этих требований были спроектированы нейронная сеть, решающая задачу классификации ста достопримечательностей, и нейронная сеть, решающая задачу классификации четырех направлений съемки. Также были разработаны диаграмма вариантов использования и макет графического интерфейса для веб-приложения.

4. РЕАЛИЗАЦИЯ

Средства разработки

Для реализации системы определения геолокации фотографии был использован высокоуровневый язык программирования Python 3.6. Разработка велась в среде разработки PyCharm Community Edition 2018.1.3 [9].

В процессе разработки были использованы следующие инструменты:

1) программные библиотеки Keras, TensorFlow, Pandas, PIL, google_streetview [10] и т.д.;

2) веб-фреймворк Flask [11];

3) Street View Static API [12] и Maps Javascript API [13] от компании Google;

4) языки HTML5 и CSS3 для верстки веб-приложения;

5) язык JavaScript и библиотека jQuery 3.3.1 для интерактивности.

4.1. Нейронная сеть для классификации достопримечательностей

4.1.1. Формирование обучающей, проверочной и тестовой выборок

Для формирования выборок были использованы данные, описанные в [14] и опубликованные в рамках соревнования Google Landmark Recognition на сайте kaggle.com. Исходные данные представлены в виде текстовых файлов с расширением csv – train.csv и test.csv. Работа с этими файлами осуществляется с использованием библиотеки Pandas [15], позволяющей работать с табличной структурой данных.

Каждому изображению в обучающей выборке присвоены уникальный id, url-адрес, а также id достопримечательности. У объектов тестовой выборки имеются id и url-адрес.

Для выделения необходимого количества классов был создан список ста самых часто повторяющихся id достопримечательностей. Функция создания данного списка представлена в листинге 1.

Листинг 1. Функция создания списка ста самых часто повторяющихся id достопримечательностей

```
def get_list_of_ids(number_of_classes, original_train_data):
      temp =
pd.DataFrame(original_train_data.landmark_id.value_counts().head(num_of_classes))
      temp.reset_index(inplace=True)
      temp.columns = ['landmark_id', 'count']
      list_of_ids = temp['landmark_id'].tolist()
return list_of_ids
```

Далее после открытия исходного файла train.csv и создания новых для записи используется цикл, в котором 80% строк из исходного файла с текущим id достопримечательности записываются в новый файл обучающей выборки, 10% отводится на проверочную выборку и еще 10% на тестовую.

4.1.2. Скачивание изображений

В связи с тем, что данные для выборок собирались из разных источников, они имеют разное разрешение. Было принято решение привести все изображения к разрешению 128x128, так как данного количества пикселей достаточно для корректного отображения достопримечательностей.

Многие url-адреса изображений представлены в формате «https://…/s1600/», что позволяет уменьшить разрешение изображений для ускорения процесса скачивания путем переписывания конца строки - «https://…/s128/». В случае, если url-адреса представлены в другом формате, они скачиваются в оригинальном разрешении.

Для приведения изображений к разрешению 128x128 была написана функция make_square, представленная в листинге 2.

Как видно из листинга, было принято решение масштабировать изображения, то есть изменять их разрешение с сохранением пропорций. Для того, чтобы разрешение изображения в итоге составляло 128x128, оно было дополнено «пустыми» пикселами, как показано на рисунке 8.

Листинг 2. Функция изменения разрешения изображения

```
def make_square(image, size=TargetSize, fill_color=(0, 0, 0, 0)):
       x, y = image.size
       max_size = max(x, y)
       k = max_size / size
       x_new, y_new = int(x / k), int(y / k)
       new_image = Image.new('RGB', (size, size), fill_color)
       image_resize = image.resize((x_new, y_new))
       new_image.paste(image_resize, (int((size-x_new)/2), int((size-
y_new)/2)))
       return new_image
```

Рис. 8. Примеры преобразованных изображений

В результате размер обучающей выборки для ста классов составил 296831 изображение, проверочной – 37000 изображений, а тестовой – 37082 изображения. В таблице 3 приведено общее количество изображений для некоторых классов. Также для каждого класса был создан подкаталог для дальнейшего использования генераторами изображений.

Табл. 3. Общее количество изображений в выборках для различных достопримечательностей

Id достопримечательности	**Количество изображений**
9633	50337
6051	50148
6599	23415
…	…
5421	1097
6347	1096
11536	1094

4.2. Нейронная сеть для определения направления съемки

4.2.1. Формирование обучающей и тестовой выборок

Для решения данной задачи было решено использовать Street View Static API (Application Programming Interface) от компании Google, позволивший создать выборки на основе панорамных фотографий, находящихся поблизости от различных достопримечательностей. Подходящие панорамы были выбраны вручную, так как достопримечательности имеют различные масштабы (например, площадь некоторых природных объектов может составлять несколько сотен квадратных километров) и они не всегда находятся на достаточно открытой местности для автоматического расчета координат панорам (например, их окружает плотная застройка, из-за которой достопримечательность не видно даже с расстояния нескольких метров). Также в некоторых случаях были вручную подобраны такие параметры, как поворот камеры по вертикали и угол поля зрения.

Всего были созданы пятнадцать наборов данных. Для каждой рассмотренной достопримечательности были скачаны изображения для четырех классов: север (поворот камеры на 0 градусов по горизонтали), юг (поворот камеры на 180 градусов по горизонтали), восток (поворот камеры на 90 градусов по горизонтали) и запад (поворот камеры на 270 градусов по горизонтали). В листинге 3 представлен пример программного кода для скачивания фотографий одной из достопримечательностей (Колонны Победы в Берлине), сделанных с запада.

Листинг 3. Пример скрипта для создания выборки на основе панорамных фотографий для одной из достопримечательностей

```
import google_streetview.api
import google_streetview.helpers
import numpy as np


key = "AIzaSyB51EvHCrqBvoHCqm1ABo0eYtLsAL7di08"
landmark_class = 4
#coordinates 52.5145, 13.3501
myloc = r"C:\Users\adeli\Desktop\google-data\final\landmark-" +
```

```
str(landmark_class)
# The landmark is in the
north = '52.5138242,13.3498604; 52.5142071,13.3500359'
east = '52.5146118,13.3494368; 52.5144564,13.3491008'
south = '52.5150997,13.3499986; 52.5150519,13.3499539'
west = '52.5145376,13.3515325; 52.514668,13.3518308'

heading_list = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5] #angles
fov_list = [90] #zoom
pitch_list = [10, 15] #up and down

location = east
if location == north:
myloc += r"\0"
    k = 0
    heading_list = list(np.asarray(heading_list) + 90 * k)
if location == east:
    myloc += r"\1"
    k = 1
    heading_list = list(np.asarray(heading_list) + 90 * k)
if location == south:
    myloc += r"\2"
    k = 2
    heading_list = list(np.asarray(heading_list) + 90 * k)
if location == west:
    k = 3
    heading_list = list(np.asarray(heading_list) + 90 * k)
    myloc += r"\3"

h = ';'.join(str(elem) for elem in heading_list)
f = ';'.join(str(elem) for elem in fov_list)
p = ';'.join(str(elem) for elem in pitch_list)

print(myloc)
apiargs = {
  'location': location,
  'size': '640x640',
  'heading': h,
  'fov': f,
  'pitch': p,
  'key': key
}
```

```
heading_list = list(np.asarray(heading_list) + 90)
api_list = google_streetview.helpers.api_list(apiargs)
results = google_streetview.api.results(api_list)
results.download_links(myloc)
```

Из листинга можно заметить, что изображения скачиваются в разрешении 640*640, однако впоследствии размер изображений в выборке был изменен на 256*256 пикселей.

Также было произведено деление данных на обучающую и тестовую выборки, где на тестовую выборку было выделено от 17% до 27% от общего количества изображений в классе для каждого из четырех классов. В таблице 4 представлено количество изображений для тестовой и обучающей выборок для каждого из пятнадцати наборов данных.

Табл. 4. Наборы данных для обучения нейронной сети для определения направления съемки

Класс достопримечательности	**Количество изображений в обучающей выборке**	**Количество изображений в тестовой выборке**
0	64	24
1	136	29
2	144	32
3	182	38
4	329	67
5	136	29
6	215	43
7	144	32
8	144	32
9	72	16
14	144	32
23	144	32
66	144	32
72	144	32
97	163	35

4.3. Реализация нейронных сетей

Обе сверточные нейронные сети были реализованы на языке Python 3.6 с помощью библиотеки Keras. В качестве вычислительного бэкенда использовался TensorFlow. Для загрузки фотографий в Keras использовались генераторы изображений (ImageDataGenerator), которые позволяют автоматически загружать изображения с диска в память компьютера, преобразовывать их в вид, необходимый Keras, и передавать в модель для обучения. Код реализации и обучения нейронной сети для классификации достопримечательностей представлен в приложении Б. Код реализации и обучения нейронной сети для определения направления съемки представлен в приложении В. Переменной landmark_class присваивается значение класса достопримечательности, данные которого используются для обучения нейронной сети. Всего данная нейронная сеть была обучена на 15 наборах данных. В рассмотренном примере нейронная сеть была обучена на данных для класса 97. Обученные нейронные сети сохраняются в каталогах, имена которых соответствуют указанному номеру класса.

4.4. Разработка веб-приложения

Для того, чтобы процесс определения геолокации изображения был максимально удобным и доступным было разработано веб-приложение, которое позволяет определять геолокацию фотографий онлайн без установки каких-либо программ.

Дизайн веб-приложения

Для веб-приложения был сверстан шаблон на HTML5 и CSS3, согласно макету графического интерфейса, представленного на рисунке 7. Для добавления интерактивных элементов был использован язык JavaScript и его библиотека jQuery. Результат представлен на рисунке 9.

Принцип работы

Для наглядной демонстрации процесса взаимодействия пользователя с клиентской частью приложения и процесса взаимодействия клиентской

части с серверной, была разработана диаграмма последовательности, представленная на рисунке 10.

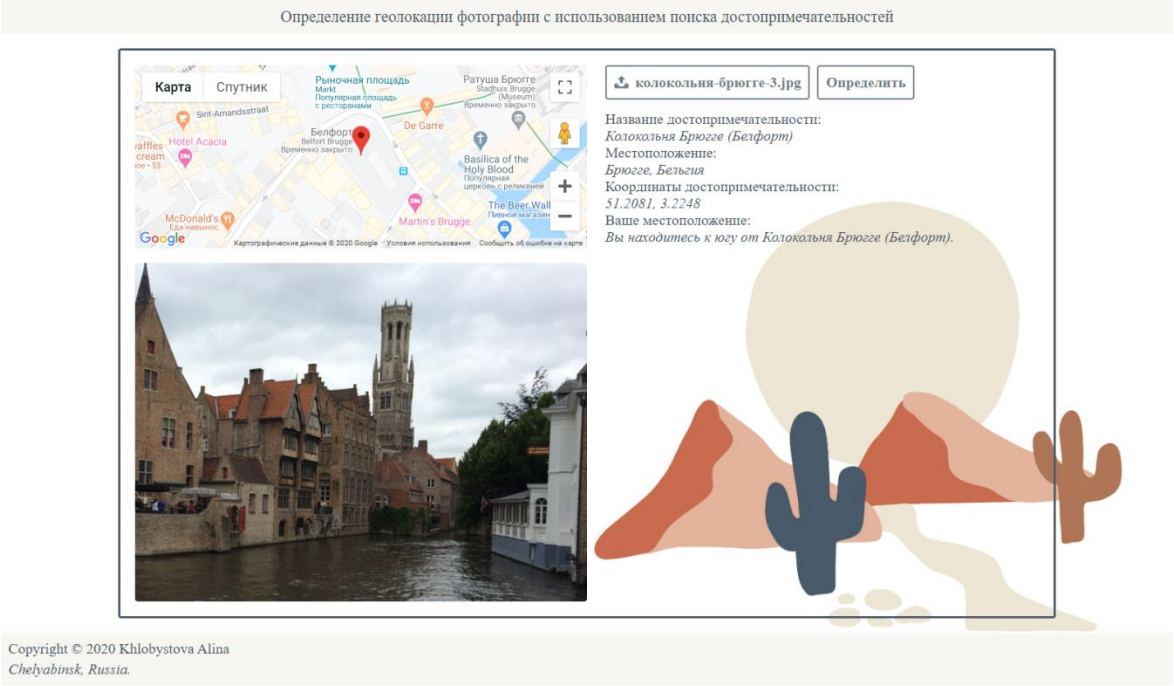

Рис. 9. Шаблон веб-приложения

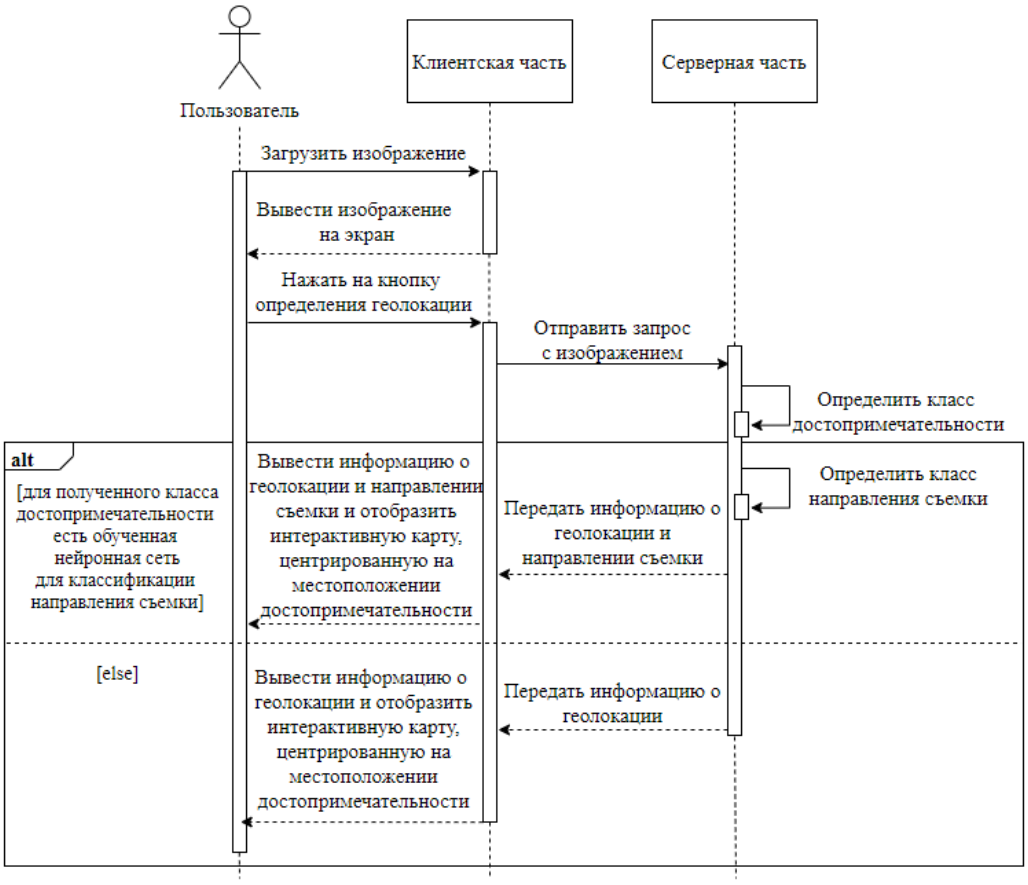

Рис. 10. Диаграмма последовательности

Для того, чтобы определить геолокацию фотографии нужно выбрать изображение в формате jpg, если изображение корректное, то оно выводится на экран, в ином случае выводится сообщение об ошибке (рисунок 11).

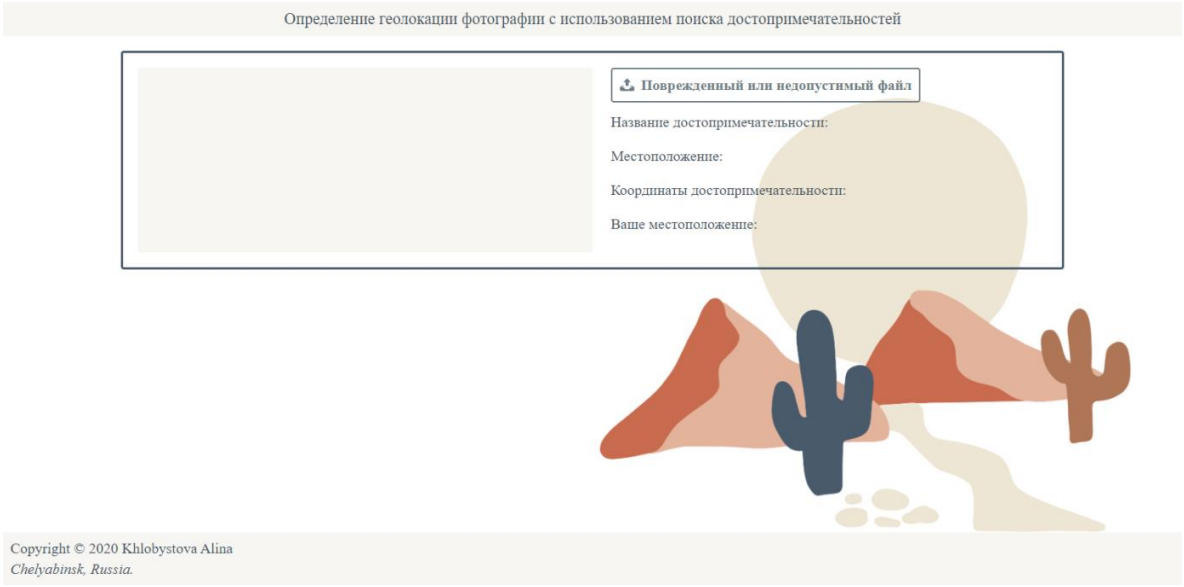

Рис. 11. Сообщение о поврежденном или недопустимом файле

При нажатии на кнопку «Определить» на сервер отправляется запрос с изображением. На сервере происходит предварительная обработка изображения: преобразование в изображение в градациях серого, изменение разрешения и конвертация в массив numpy, который можно передать нейронной сети для распознавания. Код функции, реализующий предобработку изображения, представлен в листинге 4. В данной функции используется рассмотренная ранее функция make_square (листинг 3).

Листинг 4. Функция предварительной обработки изображения.

```
def preprocess_image(image, target_size):
    image = make_square(image, target_size)
    image = image.convert("L")
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)
    return image
```

Затем происходит классификация изображения нейронной сетью для

определения класса достопримечательности, после чего в соответствии с номером класса загружается обученная модель нейронной сети для определения направления съемки для конкретной достопримечательности, если такая модель имеется. Также происходит извлечение информации о классе достопримечательности из json-файла, хранящегося в файловой системе.

В качестве ответа сервер возвращает объект json, в котором содержатся название достопримечательности, ее местоположение, координаты и информация о местоположении пользователя относительно достопримечательности. Результаты выводятся на экран.

Так как нейронная сеть для определения направления съемки была обучена лишь на 15 наборах данных, для большинства классов направление определить невозможно. В таком случае в поле «Ваше местоположение» выводится сообщение «Не удалось определить направление».

Выводы по четвертому разделу

В данном разделе была рассмотрена реализация сверточной нейронной сети для классификации достопримечательности и сверточной нейронной сети для классификации направления съемки. Также был описан процесс создания выборок для каждой из них.

Помимо этого, на основе разработанной архитектуры было реализовано приложение для определения геолокации фотографии с использованием поиска достопримечательностей с помощью нейросетевого подхода. Разработанное приложение соответствует заданным функциональным и нефункциональным требованиям.

5. ТЕСТИРОВАНИЕ

5.1. Тестирование нейронной сети для классификации достопримечательностей

Сверточная нейронная сеть для классификации достопримечательностей была обучена на 20 эпохах. В результате обучения точность на обучающей выборке составила 89,07%, точность на тестовых данных составила 83,96%.

В качестве функции потерь использовалась категориальная перекрестная энтропия (categorical_crossentropy), которая измеряет степень непохожести истинного распределения меток y и предсказанного распределения y'[16].

Для оптимизации использовался метод стохастического градиентного спуска (SGD).

5.2. Тестирование нейронной сети для классификации направления съемки

Сверточная нейронная сеть для классификации определения направления съемки была обучена на 15 наборах данных, в каждом случае на обучение отводилось 10 эпох. Результаты обучения для каждого случая представлены в таблице 5.

В качестве функции потерь так же использовалась категориальная перекрестная энтропия, а для оптимизации использовался метод стохастического градиентного спуска.

Табл. 5. Результаты обучения нейронной сети для определения направления съемки на различных наборах данных

Класс достопримечательности	Точность на обучающей выборке	Точность на тестовой выборке
0	90,46%	100%
1	94,81%	96,30%

Окончание табл. 4

Класс достопримечательности	Точность на обучающей выборке	Точность на тестовой выборке
2	88,89%	96,67%
3	78,06%	86,11%
4	78,44%	93,94%
5	73,32%	88,89%
6	92,96%	100%
7	62,50%	100%
8	83,33%	100%
9	93,06%	100%
14	96,53%	100%
23	79,86%	100%
66	90,28%	100%
72	85,42%	100%
97	84,38%	100%

Как видно из таблицы 4, выбранная архитектура сверточной нейронной сети обеспечивает хорошую точность для каждого набора данных, однако стоит заметить, что для использования полученных обученных моделей требуется большая вычислительная мощность компьютера, так как она содержит более 260 миллионов параметров.

Тестирование было выполнено на машине со следующими характеристиками:

1) центральный процессор Intel Core i7-6850K CPU @ 3.60GHz (12 CPUs), ~3.6 GHz;

2) графический процессор NVIDIA GeForce GTX 1080 8 GB GDDR5.

В большинстве случаев на загрузку одной обученной модели нейронной сети для классификации направления съемки требовалось до 1 секунды, однако в некоторых случаях этот процесс занимал около 10 секунд.

Также стоит заметить, что обучающая выборка содержит небольшое количество ракурсов и многие фотографии были сделаны в одном времен-

ном промежутке, поэтому результаты тестирования на реальных примерах отличаются. Например, для класса достопримечательности 2 точность на тестовой выборке, отличающейся от выборки, полученной путем скачивания изображений с помощью Google Street View Static API, составляет 33,33%, для класса достопримечательности 14 – 55,56%, для класса достопримечательности 66 – 50,00%.

5.3. Функциональное тестирование

Было выполнено функциональное тестирование в соответствии с функциональными требованиями, представленными в разделе 3.1. Результаты тестирования приведены ниже.

Тест №1. Запуск приложения.

Ожидаемый результат: веб-приложение должно запускаться и выводить шаблон в браузере.

Результат: работает корректно.

Тест №2. Загрузка изображения.

Ожидаемый результат: веб-приложение должно загрузить выбранное изображение, вывести его на экран и вывести имя выбранного файла в форму загрузки изображения.

Результат: работает корректно.

Тест №3. Определение геолокации фотографии.

Ожидаемый результат: веб-приложение должно начать процесс классификации изображения и по окончании данного процесса вывести информацию о геолокации фотографии, достопримечательности, присутствующей на ней и местоположении пользователя относительно достопримечательности.

Результат: работает корректно.

Тест №4. Загрузка поврежденного файла или файла с недопустимым расширением.

Ожидаемый результат: веб-приложение должно вывести ошибку

«Поврежденный или недопустимый файл» в форму загрузки изображения.

Результат: работает корректно.

Тест №5. Отображение интерактивной карты.

Ожидаемый результат: по завершению процесса классификации достопримечательности веб-приложение должно отобразить интерактивную карту, центрированную на координатах достопримечательности, помеченных маркером.

Результат: работает корректно.

Тестирование на реальных примерах

Для тестирования были подготовлены несколько изображений. Все изображения и полученные результаты приведены в приложении Г.

Выводы по пятому разделу

Было проведено тестирование разработанной модели сверточной нейронной сети, а также проведено функциональное тестирование веб-приложения для определения геолокации фотографии. Приведено тестирование приложение на реальных примерах.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы был разработан веб-сервис для геолокации фотографий достопримечательностей с помощью нейронных сетей. В ходе работы над данным проектом были решены следующие задачи:

1) проведен обзор существующих аналогов и научной литературы;

2) были подготовлены обучающая, проверочная и тестовая выборки достопримечательностей;

3) была разработана архитектура нейронной сети, занимающейся классификацией достопримечательностей;

4) были подготовлены наборы данных, состоящие из обучающей и тестовой выборок, для пятнадцати достопримечательностей;

5) была разработана архитектура нейронной сети, позволяющая определить направление съемки;

6) спроектированные нейронные сети были обучены и протестированы;

7) было разработано и протестировано приложение для определения геолокации фотографии с использованием поиска достопримечательностей.

ЛИТЕРАТУРА

1. Blippar Sightsee like never before with Blippar’s Landmark Recognition technology [Электронный ресурс]. URL: https://blippar.com/en/resources/blog/2018/02/16/landmark-recognition-api-never-confuse-landmark-again/ (дата обращения: 05.06.2019).

2. Лисовицкий А. Новый API Blippar распознает достопримечательности без GPS | Голографика [Электронный ресурс]. URL: https://holographica.space/news/blippar-api-landmark-recognition-14835 (дата обращения: 05.06.2019).

3. Weyand Google T.PlaNet-Photo Geolocation with Convolutional Neural Networks / T. Weyand Google, J. P. Google.

4. Photo Location & Online EXIF Data Viewer - Pic 2 Map [Электронный ресурс]. URL: https://www.pic2map.com/ (дата обращения: 24.03.2019).

5. TensorFlow [Электронный ресурс]. URL: https://www.tensorflow.org/ (дата обращения: 24.03.2019).

6. Welcome — Theano 1.0.0 documentation [Электронный ресурс]. URL: http://deeplearning.net/software/theano/ (дата обращения: 04.05.2019).

7. Keras Documentation [Электронный ресурс]. URL: https://keras.io/ (дата обращения: 28.02.2019).

8. Сверточные нейронные сети: взгляд изнутри — CodeSide [Электронный ресурс]. URL: http://ru.datasides.com/code/cnn-convolutional-neural-networks/ (дата обращения: 01.06.2019).

9. PyCharm: the Python IDE for Professional Developers by JetBrains [Электронный ресурс]. URL: https://www.jetbrains.com/pycharm/ (дата обращения: 20.05.2019).

10. google_streetview — google_streetview 1.2.9 documentation [Электронный ресурс]. URL: https://rrwen.github.io/google_streetview/index.html (дата обращения: 29.05.2020).

11. Welcome | Flask (A Python Microframework) [Электронный ре-

сурс]. URL: http://flask.pocoo.org/ (дата обращения: 20.05.2019).

12. Developer Guide | Street View Static API | Google Developers [Электронный ресурс]. URL: https://developers.google.com/maps/documentation/streetview/intro (дата обращения: 05.06.2020).

13. Overview | Maps JavaScript API | Google Developers [Электронный ресурс]. URL: https://developers.google.com/maps/documentation/javascript/tutorial (дата обращения: 05.06.2020).

14. Noh H. Large-Scale Image Retrieval with Attentive Deep Local Features , 2017. – 3476–3485с.

15. Python Data Analysis Library — pandas: Python Data Analysis Library [Электронный ресурс]. URL: https://pandas.pydata.org/ (дата обращения: 24.03.2019).

16. Гольдберг Й. Нейросетевые методы в обработке естественного языка / Й. Гольдберг / под ред. А.А. Слинкина. – – М.: ДМК Пресс, 2019. Вып. пер. с анг.– 282с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А – Спецификация вариантов использования

Табл. А.1. Спецификация прецедента «Выбрать фотографию»

UseCase: Выбрать фотографию
ID: 1
Аннотация: Пользователь может выбрать и загрузить фотографию в формате jpg.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: Нет
Основной поток: 1. Пользователь нажимает на кнопку для выбора фотографии. 2. Пользователь выбирает фотографию. 3. Появляется кнопка «Определить». 4. Фотография выводится на экран.
Постусловия: Нет
Альтернативные потоки: 1. Пользователь выбирает поврежденный или недопустимый файл. 2. В форме загрузки изображения выводится сообщение об ошибке.

Табл. А.2. Спецификация прецедента «Определить геолокацию фотографии»

UseCase: Определить геолокацию фотографии
ID: 2
Аннотация: Пользователь запускает процесс распознавания изображения.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: Выбрано корректное изображение в формате jpg.

Окончание табл. А.2.

UseCase: Определить геолокацию фотографии
Основной поток: 1. Пользователь нажимает на кнопку «Определить». 2. Выводится информация о геолокации фотографии, включая информацию об относительном местоположении пользователя.
Постусловия: Нет
Альтернативные потоки: 1. Для полученного в результате классификации класса достопримечательности нет обученной нейронной сети, определяющей направление съемки. В поле «Ваше местоположение» выводится текст «Не удалось определить направление».

Табл. А.3. Спецификация прецедента «Взаимодействовать с интерактивной картой»

UseCase: Взаимодействовать с интерактивной картой
ID: 3
Аннотация: После того, как был определен класс достопримечательности, на экране отображается интерактивная карта, центрированная на координатах достопримечательности, которые помечаются маркером. Пользователь может выбрать режим отображения карты («Карта» или «Спутник»), перемещаться по ней, менять масштаб, переключаться в полноэкранный режим, а также режим просмотра улиц.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: Был определен класс достопримечательности, присутствующей на загруженном изображении.
Основной поток: 1. Пользователь использует интерактивную карту.
Постусловия: Нет
Альтернативные потоки: Нет

ПРИЛОЖЕНИЕ Б – Реализация и обучение нейронной сети для классификации достопримечательностей

Листинг Б.1. Код реализации и обучения нейронной сети для классификации достопримечательностей

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
import keras
import numpy as np
import os, datetime, sys
import tensorflow as tf

np.random.seed(42)

train_dir = 'train_100_classes_128_size'
test_dir = 'test_100_classes_128_size'
val_dir = 'val_100_classes_128_size'
img_w, img_h = 128, 128
input_shape = (img_w, img_h, 1)
epochs = 20
batch_size = 20

model = Sequential()

model.add(Conv2D(32, (4, 4), input_shape=input_shape,
       padding='same', activation='relu'))
model.add(Conv2D(32, (4, 4), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(64, (4, 4), padding='same', activation='relu'))
model.add(Conv2D(64, (4, 4), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(100, activation='softmax'))
```

```
model.add(Dropout(0.5))
model.compile(loss='categorical_crossentropy', optimizer='SGD',
      metrics=['accuracy'])

train_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator(rescale=1. / 255)
val_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(train_dir,
      target_size=(img_w, img_h), batch_size=batch_size,
      class_mode='categorical', color_mode='grayscale')

test_generator = test_datagen.flow_from_directory(test_dir,
      target_size=(img_w, img_h), batch_size=batch_size,
      class_mode='categorical', color_mode='grayscale')

val_generator = val_datagen.flow_from_directory(val_dir,
      target_size=(img_w, img_h), batch_size=batch_size,
      class_mode='categorical', shuffle=False, color_mode='grayscale')

nb_train = train_datagen.flow_from_directory(train_dir,
      target_size=(img_w, img_h), batch_size=1,
      class_mode='categorical').__len__()
nb_val = val_datagen.flow_from_directory(val_dir,
      target_size=(img_w, img_h), batch_size=1,
      class_mode='categorical').__len__()
nb_test = test_datagen.flow_from_directory(test_dir,
      target_size=(img_w, img_h), batch_size=1,
      class_mode='categorical').__len__()

checkpoint_path = "training_1/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

if not os.path.exists(checkpoint_dir):
    os.mkdir(checkpoint_dir)

cp_callback = keras.callbacks.ModelCheckpoint(checkpoint_path,
      save_weights_only=True, verbose=1)

model.fit_generator(train_generator, steps_per_epoch=nb_train // batch_size,
      epochs=epochs, validation_data=val_generator,
validation_steps=nb_val // batch_size, callbacks=[cp_callback])
```

```
scores = model.evaluate_generator(test_generator, nb_test // batch_size)
now_time = datetime.datetime.now()
str_cur_time = '{}_{}_{}_{}_{}_{}'.format( now_time.year, now_time.month,
      now_time.day, now_time.hour, now_time.minute, now_time.second )

save_model_filepath = 'models/grayscale_model_score_' + str(scores[1]*10000.0
// 100) + '%_' + str_cur_time + '.h5'

model.save(save_model_filepath)

print("Точность на тестовых данных: %.2f%%" % (scores[1]*100))
```

ПРИЛОЖЕНИЕ В – Реализация и обучение нейронной сети для классификации направления съемки

Листинг В.1. Код реализации и обучения нейронной сети для классификации направления съемки

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
import keras
import numpy as np
import os, datetime, sys
import tensorflow as tf

np.random.seed(42)
landmark_class = 97

train_dir = 'data/landmark-' + str(landmark_class) + '-r/train'
test_dir = 'data/landmark-' + str(landmark_class) + '-r/test'
img_w, img_h = 256, 256
input_shape = (img_w, img_h, 1)
epochs = 10
batch_size = 3

model = Sequential()

model.add(Conv2D(64, (4, 4), input_shape=input_shape,
      padding='same', activation='relu'))
model.add(Conv2D(64, (4, 4), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(128, (4, 4), padding='same', activation='relu'))
model.add(Conv2D(128, (4, 4), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(128, activation='relu'))
```

```
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='SGD',
      metrics=['accuracy'])

train_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(train_dir,
      target_size=(img_w, img_h), batch_size=batch_size,
      class_mode='categorical', color_mode='grayscale')

test_generator = test_datagen.flow_from_directory(test_dir,
      target_size=(img_w, img_h), batch_size=batch_size,
      class_mode='categorical', color_mode='grayscale')

nb_train = train_datagen.flow_from_directory(train_dir,
      target_size=(img_w, img_h), batch_size=1,
      class_mode='categorical').__len__()
nb_test = test_datagen.flow_from_directory(test_dir,
      target_size=(img_w, img_h), batch_size=1,
      class_mode='categorical').__len__()

checkpoint_path = "training_cp/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

if not os.path.exists(checkpoint_dir):
    os.mkdir(checkpoint_dir)

cp_callback = keras.callbacks.ModelCheckpoint(checkpoint_path,
      save_weights_only=True, verbose=1)

model.fit_generator(train_generator, steps_per_epoch=nb_train // batch_size,
      epochs=epochs, callbacks=[cp_callback])

scores = model.evaluate_generator(test_generator, nb_test // batch_size)

now_time = datetime.datetime.now()
str_cur_time = '{}_{}_{}_{}_{}_{}'.format(now_time.year, now_time.month,
now_time.day, now_time.hour, now_time.minute, now_time.second)
save_model_filepath = 'models/' + str(landmark_class) +
```

```
'/grayscale_model_score_' + str(scores[1]*10000.0 // 100) + '%_' + str_cur_time
+ '.h5'
models_dir = os.path.dirname(save_model_filepath)

if not os.path.exists(models_dir):
    os.mkdir(models_dir)

model.save(save_model_filepath)

print("Точность на тестовых данных: %.2f%%" % (scores[1]*100))
```

ПРИЛОЖЕНИЕ Г – Тестирование веб-сервиса на реальных примерах

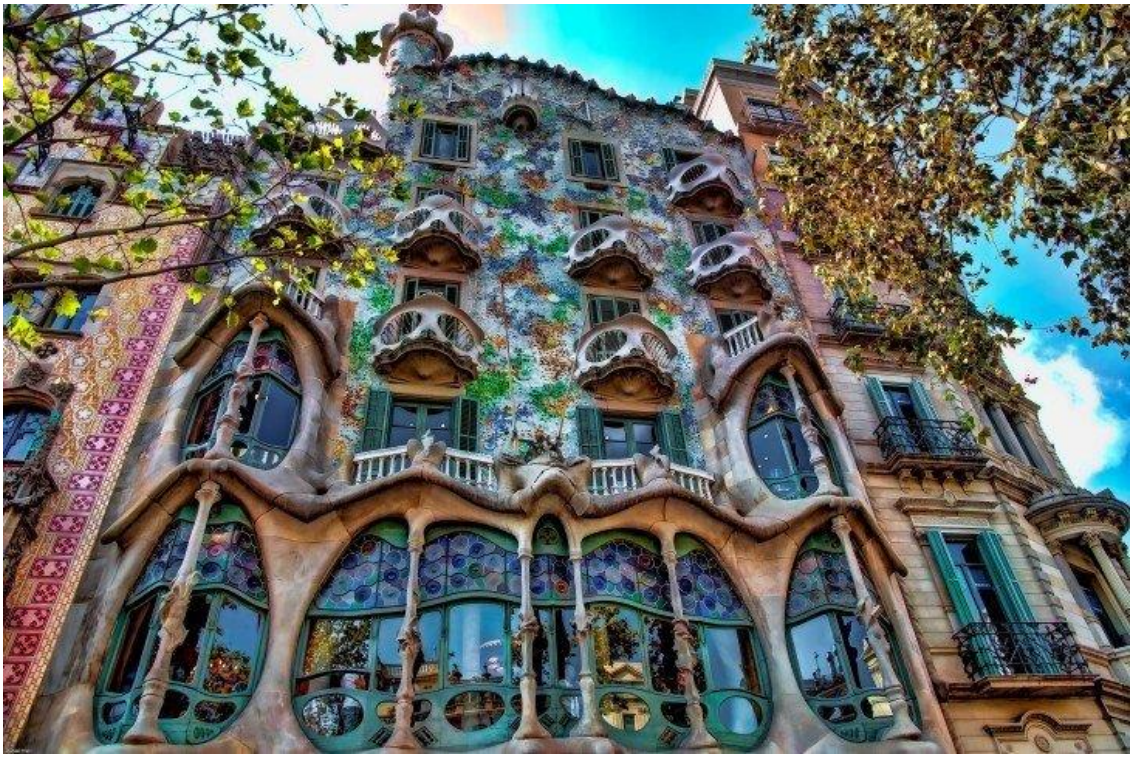

Рис. Г.1. Дом Бальо, Испания

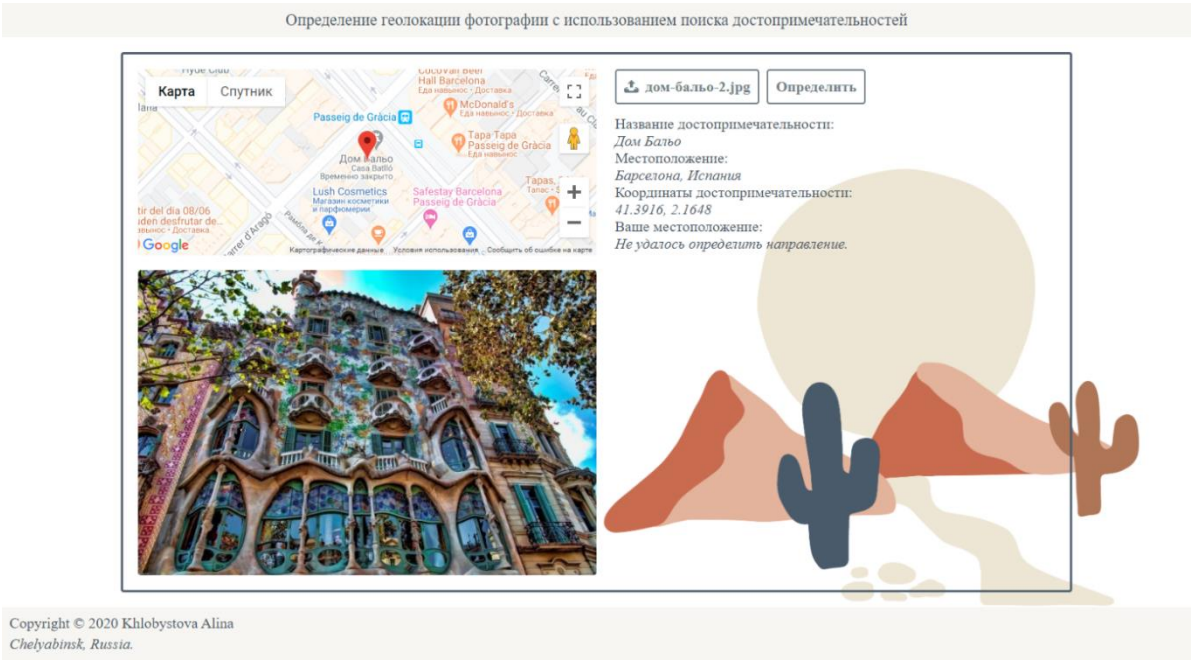

Рис. Г.2. Результат работы веб-сервиса (обученная модель нейронной сети для классификации направления съемки отсутствует)

Рис. Г.3. Хофкирхе, Германия, вид со стороны востока

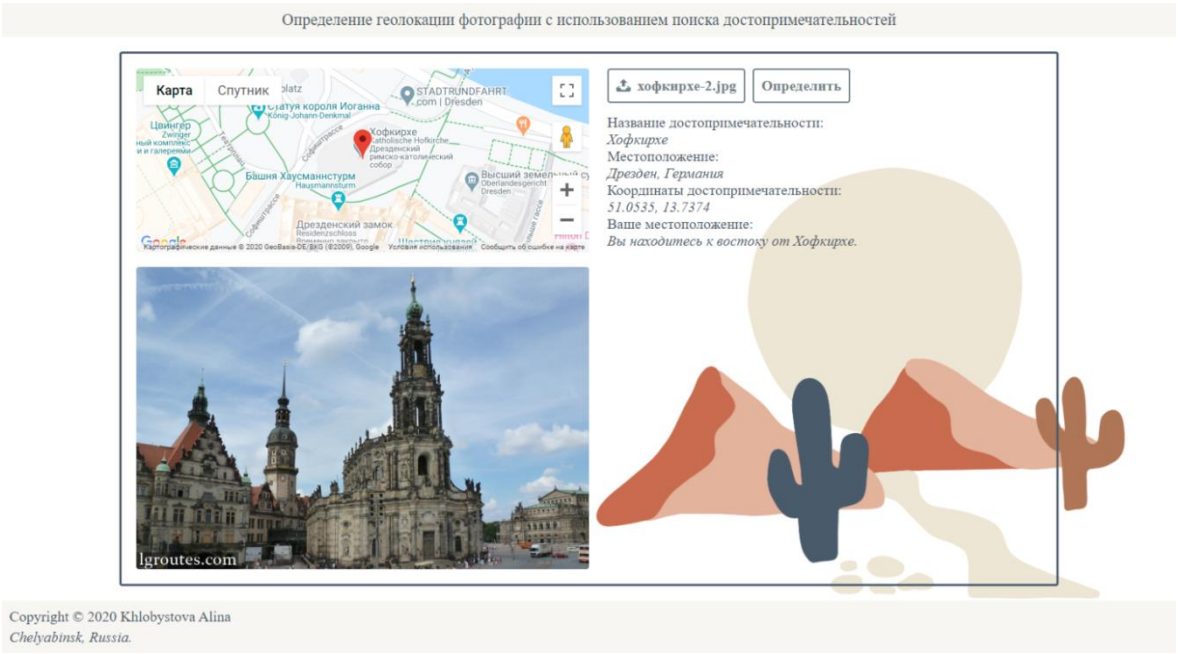

Рис. Г.4. Результат работы веб-сервиса

Рис. Г.5. Колизей, Италия, вид со стороны юга

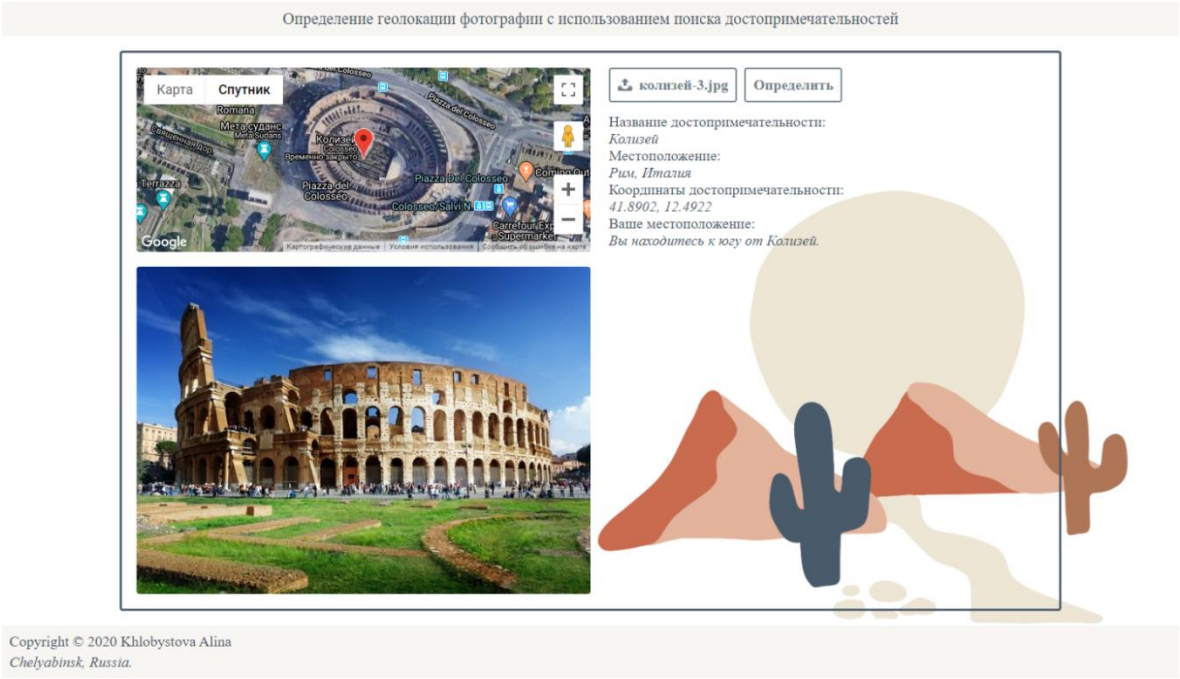

Рис. Г.6. Результат работы веб-сервиса, интерактивная карта в режиме «Спутник»