МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
Национальный исследовательский ядерный университет «МИФИ»

Институт

интеллектуальных кибернетических систем

Кафедра кибернетики (№ 22)

Направление подготовки 09.03.04 Программная инженерия

Пояснительная записка

к учебно-исследовательской работе студента на тему:

Разработка метода геолокации по серии изображений на основе глубокого обучения

Группа Б14-506

Студент _____ Шедько А.Ю.
(подпись) (ФИО)

Руководитель _____ _____ Трофимов А.Г.
(0-15 баллов) (подпись) (ФИО)

-

Научный консультант _____ _____ _____
(0-15 баллов) (подпись) (ФИО)

Оценка руководителя _____ Оценка комиссии _____

Члены комиссии

_____ _____

_____ _____

_____ _____

_____ _____

Москва 2017

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Федеральное государственное бюджетное образовательное учреждение высшего образования
Национальный исследовательский ядерный университет «МИФИ»

Факультет Кибернетики и Информационной безопасности

Кафедра кибернетики (№ 22)

Направление подготовки 09.04.04 Программная инженерия

Пояснительная записка

к ВКР магистра на тему:

_____

_____

Группа _____

Студент _____ (подпись) _____ (ФИО)

Руководитель _____ (подпись) _____ (ФИО)

Научный консультант _____ (подпись) _____ (ФИО)

Оценка руководителя _____ Оценка комиссии _____

Москва 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем
КАФЕДРА КИБЕРНЕТИКИ

Задание на УИР

Студенту гр. Б14-506 (группа) — Шедько Андрею Юрьевичу (фио)

ТЕМА УИР

Разработка метода геолокации по серии изображений на основе глубокого обучения

ЗАДАНИЕ

№ п/п	Содержание работы	Форма отчетности	Срок исполнения	Отметка о выполнении
1.	**Аналитическая часть**			
1.1.	Обзор методов геолокации по изображениям	Текст ПЗ	18.03.2018	
1.2.	Изучение и сравнительный анализ алгоритмов глубокого обучения с целью выбора подхода к задаче	Текст ПЗ	18.03.2018	
1.3.	Анализ алгоритмов пространственного разбиения поверхности земли для решения задачи классификации	Текст ПЗ	18.03.2018	
1.4.	Анализ существующих решений задачи геолокации по изображениям.	Текст ПЗ	18.03.2018	
1.5.	Анализ возможностей применения подхода transfer learning к проблеме геолокации с помощью глубокого обучения	Текст ПЗ	18.03.2018	
2.	**Теоретическая часть**			
2.1.	Формальная постановка задачи геолокации по серии изображений	Текст ПЗ	18.03.2018	
2.2.	Выбор/разработка методов оценки точности работы алгоритмов геолокации	Метод	18.03.2018	
2.3.	Модификация существующих решений в области для работы с серией изображений	Алгоритм	18.03.2018	
2.4.	Разработка метода геолокации по серии изображений используя выбранные/разработанные выше алгоритмы/методы	Метод	18.03.2018	
3.	**Инженерная часть**			
3.1.	Разработать архитектуру для системы (с учетом требований к области применения)	Схемы, Диаграммы	25.03.2018	
3.2.	Проектирование системы геолокации по серии изображений	Схемы, Диаграммы	25.03.2018	
3.3.	Результаты проектирования оформить с помощью диаграмм, схем, описаний. При проектировании использовать язык UML	Схемы, Диаграммы	25.03.2018	
4.	**Технологическая и практическая часть**			
4.1.	Реализовать разработанные алгоритмы	Исполняемые файлы, исходный текст	25.03.2018	

4.2.	Протестировать систему с помощью сравнения с аналогами. Разработать тестовые примеры для подтверждения исполнения требований.	Исполняемые файлы, исходные тексты тестов и тестовых примеров	Практика	
4.3.	Реализация должна показывать результат лучше чем аналоги на территории РФ	Графики, Таблицы	Практика	
4.4.	Ожидаемым результатом является программное обеспечение позволяющее осуществлять распознавание континента (2500 km), страны(750 km), города (25 km), где сделано фото	Графики, Таблицы, Исполняемые файлы	Практика	
4.5.	При реализации использовать ЯП python и библиотеки keras, tensorflow	Код программы	25.03.2018	
5.	*Оформление пояснительной записки (ПЗ) и иллюстративного материала для доклада*.	Текст ПЗ, презентация	25.03.2018	

ЛИТЕРАТУРА

1.	Weyand T., Kostrikov I., Philbin J. Planet-photo geolocation with convolutional neural networks //European Conference on Computer Vision. – Springer, Cham, 2016. – С. 37-55.
2.	Babenko A. et al. Neural codes for image retrieval //European conference on computer vision. – Springer, Cham, 2014. – С. 584-599.
3.	Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks //Advances in neural information processing systems. – 2012. – С. 1097-1105.
4.	Hays J., Efros A. A. IM2GPS: estimating geographic information from a single image //Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. – IEEE, 2008. – С. 1-8.
5.	Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. М.// ДМК, 2007
6.	Hastie, Trevor, Tibshirani, Robert and Friedman, Jerome. The elements of statistical learning: data mining, inference and prediction – 2 edition – Springer, 2009.
7.	Hays J., Efros A. A. Large-scale image geolocalization //Multimodal Location Estimation of Videos and Images. – Springer, Cham, 2015. – С. 41-62.
8.	Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей // Питер Спб – 2018 – С. 480, ISBN: 9785496025362
9.	
10.	

Дата выдачи задания: Руководитель _____ Трофимов А.Г.
(ФИО)

« ___ » февраля 2018г. Студент _____ Шедько А.Ю.
(ФИО)

Реферат

Пояснительная записка содержит 43 страницы (из них 6 страниц приложений). Количество использованных источников — 23. Количество приложений — 2. Количество иллюстраций — 16.

Ключевые слова: визуальная локализация, геолокация, глубокое обучение, нейронные сети, LSTM, свёрточные сети

Целью данной работы является разработка системы визуальной локализации на основе глубокого

В первой главе проводится обзор и анализ геолокации по изображениям, исторических решений задачи визуальной локализации, алгоритмов разбиения земной поверхности, существубщих библиотек для глубокого обучения.

Во второй главе описываются использованные и модифицированные алгоритмы классификации, архитектуры нейросетей, способы оценки точности классификации, функции потерь.

В третьей главе приводится описание программной реализации.

В четвёртой главе описаны результаты экспериментальной проверки, приводятся иллюстрации, описания результатов.

В приложении А находятся фргменты исходных текстов разработанного ПО

В приложении Б находятся дополнительные иллюстрации.

Содержание

Введение

Задача геолокации по изображениям имеет множество приложений в различных сферах человеческой деятельности. На различных масштабах это могут быть:

- ориентация в помещениях,
- автоматизированное определение местоположения по фотографии без геотегов,
- помощь в ориентации в незнакомой среде.

Актуальность данной работы проявляется в:

- Проблематика данной области интересна в широком круге применений, таких как: поиск расположения определённого объекта, автоматическое отслеживание преступника по фотографии с места проишествия , теоретические исследования алгоритмов глубокого обучения для многоклассовой пространственной классификации. Выиграют от решения этой проюлемы специальные службы и корпорации, занимающиеся подобными вещами.
- Задача ”Визуальной локализации” [1], поставленная в области компьютерного зрения одной из первых оказалась непростой как для компьютеров, так и для людей. Первой была работа У. Б. Томпосна (Университет Юты) и других [2], основанная на распознавании деталей ландшафта и сопоставлении их с картой местности. Однако при наличии объёмных обучающих выборок ситуация значительно улучшается. Современные работы Т.Вейанда (Google) и И.Кострикова (RWTH Aachen) [3] показывает значительные успехи в данной области.

Все предыдущие работы используют значительные объёмы данных и вычислительные мощности, а потому интересна задача решения подобных задач на менее производительном железе, что может быть полезно для развёртывания этой технологии в рамках мобильного приложения.

Новизна работы состоит в использовании для решения задачи современных методов машинного обучения, применения нового обучающего набора и методе его получения.

Суть исследования — описание и разработка алгоритма геолокации на основе набора изображений.

1. Анализ проблематики задачи геолокации по изображениям

1.1 Обзор методов геолокации по изображениям

Задача определения места съемки фотографии довольно непроста из-за неоднозначности и недостаточности информации, содержащейся в одном изображении. Например, типичная пляжная сцена (море, солнце, песок, небо...) может быть заснята почти в любой точке земли. Даже достопремечательности не всегда могут служить абсолютными ориентирами: Эйфелева башня может указывать на Париж с Елисейскими полями, а может на Лас-Вегас или на село Париж в Челябинской области. В отсутствие подобных ориентиров люди полагаются на такие признаки как язык дорожных знаков, разметку, окружающую флору; опираясь на знания о внешнем мире для уточнения оценки местоположения. Традиционные системы компьютерного зрения часто не обладают подобными сведениями, полагаясь лишь на то что можно почерпнуть из тестовой выборки.

Для решения задачи геолокации применялись:

- расстояния между изображениями вычисляемые при помощи глобальных дескрипторов изображений. Рассматривалось в 2008 году в работе `IM2GPS` Хейса и Эфроса из Карнеги Меллон [1]. Рассматривается далее более подробно.
- Дополнение данными спутниковой съемки [4].
- Распознавание ориентиров [5].
- Распознавание формы городского горизонта Skyline2GPS [6]

1.2 Изучение и сравнительный анализ алгоритмов Выделения признаков изображений с целью выбора подхода к задаче

Для задач обработки изображений используется множество различных методов. Перечислим некоторые из них.

1.2.1 Глобальные дескрипторы

Глобальные дескрипторы описывают изображение в целом и представляют его в виде векторов признаков. Обычно каждая точка вносит вклад в значение дескриптора. При поиске по коллекции цветных изображений для человека одной из наиболее значимых характеристик является цвет. К тому же он инвариантен относительно расположения объектов и размера изображения, что упрощает его анализ. Самое распространенное представление цвета - это цветовая гистограмма

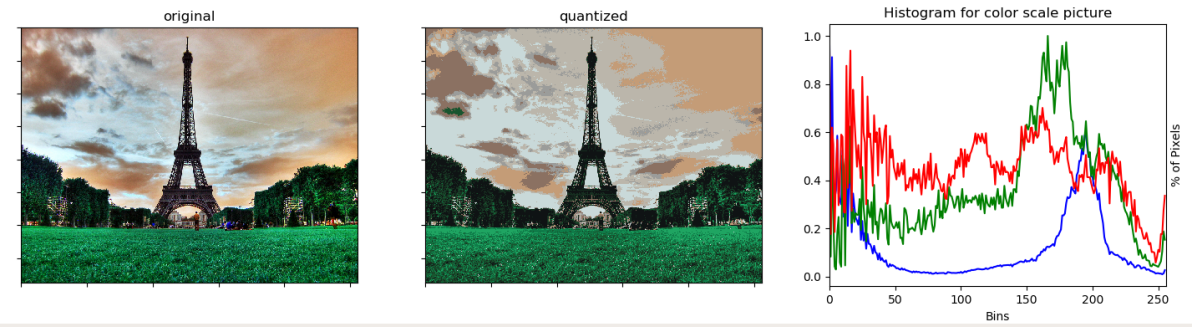


Рис. 1.1 – Гистограмма цветов и квантованная картинка.
Квантование проведено с помощью 10-NN кластеризации

[7] (гистограмма распределения цветов). Подход заключается в том, что цветовое пространство разбивается на промежутки и для каждого промежутка вычисляется доля пикселей из данного промежутка. Процесс разбиения цветового пространства на ограниченное количество цветовых диапазонов называется квантованием. Сложность данного подхода заключается в построении такого разбиения, чтобы цвета из одного диапазона были плохо различимы человеком, а цвета из разных диапазонов, наоборот, различались. Пример гистограммы с квантованием в цветовом пространстве RGB изображен на Рисунке 1.1.

В простейшем случае, и в частности в данном примере, не учитывается пространственное расположение цветов на изображении. Решением является разбиение изображения на фиксированные блоки и вычисление цветовой гистограммы отдельно для каждого блока. Однако в данном решении возникает проблема в подборе размеров блоков. Еще одним решением проблемы учета пространственного расположения цветов является модификация классической цветовой гистограммы HistSP. Главная идея данного метода заключается в том, что для каждого ненулевого элемента гистограммы вычисляется центр масс пикселей соответствующего цвета, его координаты сохраняются в числе элементов вектора признаков.

Альтернативная модель представления цвета изображения – это цветовые моменты, предложенные Stricker M., Orengo M. [8]. Авторы рассматривают распределения отдельных цветовых каналов как части трехмерного распределения. Вводится пять фиксированных областей: центральная область в виде эллипса и четыре боковые области. Для каждой области вычисляется математическое ожидание по каждому из цветовых каналов и попарные ковариации распределений каналов. Для каждого из пикселей вычисляется его степень принадлежности к области: чем ближе к границе области, тем меньше степень принадлежности к ней. Это значение регулирует вклад цвета соответствующего пикселя в общую оценку распределения цвета области. Еще одной значимой характеристикой изображения является текстура. Она описывает структуру объектов на

изображении, и определяется по распределению уровня яркости изображения.

Одним из наиболее известных представлений текстурных характеристик изображения являются матрицы смежности (матрицы совместного распределения яркости на изображении - Grey Level Co-occurrence Matrices, GLCM) [9]. Матрица смежности зависит от разницы яркости в соседних пикселях. По ней можно вычислить значения различных статистических показателей, например, таких как энтропия (степень неоднородности), контраст, показатель однородности, показатель гладкости и др. Еще одними из известных текстурных признаков являются признаки Тамуры[10], которые были выделены с учетом особенностей зрительного восприятия человека. Они включают в себя зернистость, контрастность, направленность, линейность, регулярность и грубость. Из них были определены три наиболее подходящих при решении задачи поиска изображений – это зернистость, контрастность и направленность. Если собрать эти значения в одно изображение, в котором red, green, blue каналы будут заменены на зернистость, контрастность и направленность соответственно, то из полученного изображения можно вычислить 3d текстурную гистограмму. Для рассмотрения текстуры изображения в различных масштабах, можно использовать вейвлет-анализ, который заключается в разложении сигнала по базисным функциям. Базисные функции (вейвлеты) строятся на основе порождающего вейвлета с помощью сдвига и масштабирования. Берется исходное изображение и строится первая проекция сигнала (свертка с первой базисной функцией), потом вычисляется разность исходного сигнала с полученным и строится вторая проекция этой разности (свертка со второй базисной функцией), и т.д. Причем, каждая базисная функция является сдвигом предыдущей, растянутой в $2n$ раз (n характеризует масштаб). Таким образом, в итоге получаем грубую версию изображения. Такие базисные функции обычно называют фильтрами.

Одними из эффективных и используемых фильтров являются фильтры Габора , ICA-фильтры (Independent Component Analysis, ICA [10]). ICA-фильтры получены путем анализа обучающего множества изображений. Данные фильтры являются локальными и подобны фильтрам Габора, однако в отличие от них носят естественный характер и отражают основные направления текстуры изображений, по которым они строились. Также проводились исследования, показывающие, что способ построения ICA- фильтров схож с процессом зрительного обучения человека. Еще одним важным признаком для сравнения изображений является форма объектов. Простейшими признаками являются центр тяжести фигур, площадь, направление главной оси и т.д. Существуют и более сложные методы, представляющие фигуры более детально, их можно разделить на два класса:

– внешнее представление, основанное на информации о контуре фигуры – дескрипторы границ, к ним относятся разные виды сигнатур, дескрипторы Фурье, вейвлет-дескрипторы, цеп-

ные коды и т.д.;

– внутреннее представление, основанное на информации о фигуре в целом – дескрипторы областей, к ним относятся, например, инварианты моментов и т.д.

1.2.2 Локальные дескрипторы

Локальные дескрипторы представляют собой вектора признаков, построенные по отдельным фрагментам изображения. То есть они не описывают все изображение, а содержат информацию только о выбранных некоторым способом фрагментах. Самыми известными локальными дескрипторами являются SIFT [11] (Scale Invariant Feature Transform), SURF [12] (Speeded Up Robust Features), PCA-SIFT [13] (PCA – Principal Component Analysis) и т.д. Метод SIFT (Scale Invariant Feature Transform) обнаруживает и описывает локальные особенности изображения. Получаемые с помощью него признаки инвариантны относительно масштаба и поворота, устойчивы к ряду аффинных преобразований, шуму, изменению в освещении. Данный алгоритм можно разделить на две части: определение «точек интереса» (key points, points of interest, salient points) и построение дескрипторов окрестностей данных точек. Существует несколько способов определения точек интереса. Алгоритм, предложенный в рамках SIFT, один из самых известных. Он заключается в использования пирамиды Гаусса, которая строится для изображения.

Далее изображения приводятся к одному размеру, и вычисляется их разность (DoG, difference-of-Gaussian images). Причем в качестве кандидатов точек интереса выбираются только те пиксели, которые сильно отличаются от остальных, это делается, например, путем сравнения каждого пикселя изображения с несколькими соседними данного масштаба, с несколькими соответствующими соседями в большем и меньшем масштабе. Далее для каждой такой точки интереса вычисляется локальный дескриптор, характеризующий направление градиентов в пикселях некоторой окрестности. Главным минусом SIFT дескрипторов является их высокая размерность и большое количество на изображении. PCA-SIFT [13] (PCA, Principal Component Analysis – анализ главныхкомпонент) дескриптор – одна из вариаций SIFT, в которой уменьшается размерность дескриптора с помощью анализа главных компонент. Это достигается с помощью нахождения пространства собственных векторов, на которое впоследствии проецируются вектора признаков.

Альтернативным подходом является SURF (Speeded Up Robust Features), который в несколько раз быстрее SIFT. В данном подходе для ускорения поиска точек интереса используются интегральные изображения. Значение в каждой точке интегрального изображения вычисляется как сумма значения в данной точке и значений всех точек, которые находятся выше и левее данной. С помощью интегральных изображений за константное время вычисляются так называемые прямо-

угольные фильтры, которые состоят из нескольких прямоугольных областей. SURF в несколько раз быстрее SIFT, менее чувствителен к шуму, к повороту, но чувствителен к изменению освещения или угла, под которым был сделан снимок. Глобальные и локальные дескрипторы обычно используются для решения разных задач. Глобальные дескрипторы в основном применяются для решения задачи общего поиска изображений по содержанию, то есть для поиска по запросу-образцу визуально и семантически похожих изображений. В данном случае важно все изображение в целом, а не отдельные его области, поэтому для решения данной задачи подходят глобальные дескрипторы, характеризующие все изображение. Локальные дескрипторы обычно применяются для решения задачи поиска нечетких дубликатов. Дубликатами считаются изображения одной и той же сцены или объекта, сделанные в разных условиях или разного качества, в частности, изображения одной и той же сцены в разном масштабе или снятые с разных точек, с различием в освещении или с незначительными изменениями фона. При решении данной задачи важно обнаружить сходство отдельных частей изображений, для данных целей обычно применяются локальные дескрипторы, описывающие особенности областей изображений.

Все вышеописанные локальные дескрипторы используют общую парадигму, заключающуюся в нахождении точек интереса и построении для каждой из них дескриптора, описывающего ее окрестность. Принципиально другой подход описан в работе [14]. Он основан на свойстве повторяемости (самоподобии) фрагментов на изображении, то есть на наблюдении, что небольшие фрагменты изображения имеют свойство повторяться на нем в том же или другом масштабе. Информация о такой повторяемости в пределах некоторой области изображения формирует так называемую геометрическую разметку. С помощью данной геометрической разметки формируются самоподобные локальные дескрипторы. Причем, даже если изображения имеют разную текстуру, цвет и др., но их геометрические разметки похожи, то дескрипторы тоже будут похожи. Эксперименты, проведенные авторами работы, показали применимость данных дескрипторов для решения задач распознавания объектов и поиска фрагментов в коллекциях изображений и видео без предварительного обучения. Также стоит заметить, что большинство других локальных дескрипторов строятся только для точек интереса и, следовательно, не описывают все изображение в целом. Самоподобные локальные дескрипторы изначально строятся для каждой точки изображения, после чего производится фильтрация, даже после которой самоподобные дескрипторы образуют более плотное множество по сравнению с другими описанными локальными дескрипторами.

1.2.3 Глубокие сверточные нейросети (ГСНС)

Часто для выделения значимых признаков в изображениях требуются экспертные знания, а потому их непросто описать алгоритмически. Функции выделения в ГСНС автоматически формируют изображения для определенных областей, не используя никакие функции обработки отличительных признаков. Благодаря этому процессу ГСНС пригодны для анализа изображений:

ГСНС обучают сети с множеством слоев.

- Несколько слоев работают вместе для формирования улучшенного пространства отличительных признаков.
- Начальные слои изучают первостепенные признаки (цвет, края и пр.).
- Дальнейшие слои изучают признаки более высокого порядка (в соответствии с входным набором данных).
- Наконец, признаки итогового слоя подаются в слои классификации.

Теперь осталось только формально определить, что же такое свертка и как устроены слои сверточной сети. Свертка — это всего лишь линейное преобразование входных данных особого вида. Если x^l — карта признаков в слое под номером l, то результат двумерной свертки с ядром размера $2d+1$ и матрицей весов W размера $(2d+1)\times(2d+1)$ на следующем слое будет таким:

$$y^l_{i,j} = \sum_{-d\leqslant a,b\leqslant d} W_{a,b} x^l_{i+a,j+b},$$

где $y^l_{i,j}$— результат свёртки в i,j-том пикселе на уровне l, а $x^l_{i,j}$ - её вход (выход предыдущего слоя). Иначе говоря, чтобы получить компоненту (i,j) следующего уровня, мы применяем линейное преобразование к квадратному окну предыдущего уровня, то есть скалярно умножаем пикселы из окна на вектор свертки.

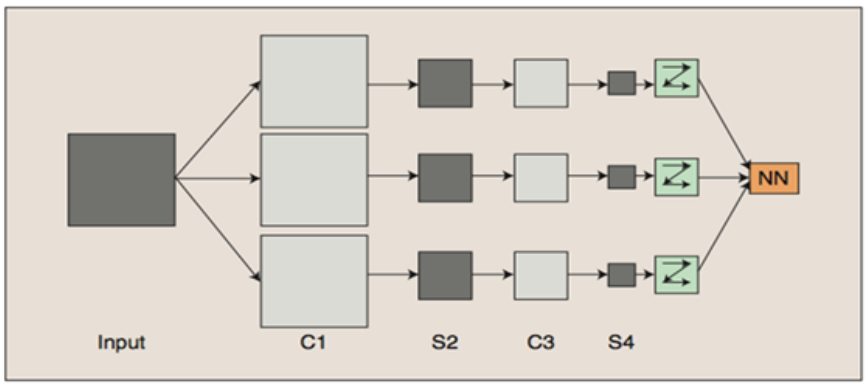


Рис. 1.2 – Пример архитектуры свёрточной сети. Слои C — свертки, слои S — пулы и выборки.

Определение 1 (Свертка)**.** Сверточные слои состоят из прямоугольной сети нейронов. Веса при

этом одинаковы для каждого нейрона в сверточном слое. Веса сверточного слоя определяют фильтр свертки.

Определение 2 (Опрос)**.** Опрашивающий слой берет небольшие прямоугольные блоки из сверточного слоя и проводит подвыборку, чтобы сделать из этого блока один выход.

1.3 Анализ алгоритмов пространственного разбиения поверхности земли для решения задачи классификации

Подразумевается что мы рассматриваем задачу геолокации как задачу классификации областей Земной поверхности, а не как задачу регрессии координат. Это позволяет работать с выходом модели как с гистограммой распределения вероятности нахождения фотографии в точках поверхности земли. Достоинство такого подхода схоже с задачей классификации изображений, когда правильный класс может быть не первым, а например одним из 5 наиболее вероятных догадок, что может быть использовано. Существует множество способов проекции 2-сферы на плоскость, однако у каждого из них есть свои недостатки: Проекция Меркатора, представитель класса равноугольных проекций, не сохраняет пропорций (Гренландия значительно меньше Африки), Равновеликие проекции не сохраняют форму объектов, что менее критично но также нежелательно. Поэтому интересным представляется следующее решение проблемы — использование не проекции на плоскость, а разбиения поверхности на области при помощи проекции поверхности на грани куба, См рис 1.3. Очевидно что если просто взять максимально плотное разбиение поверхности не даст нужного результата: будет слишком много пустых областей и областей с недостаточным числом примеров.

1.4 Существующие решения задачи геолокации по изображениям

Описанные выше в пункте по работы [3] представляют собой state-of-the-art на текущий момент, однако автору не удалось обнаружить в свободном доступе ПО, демонстрирующее функциональность описанных методов.

1.4.1 ПО для работы с алгоритмами глубокого обучения

На настоящий момент существует значительное число библиотек глубокого обучения, поэтому ограничимся лишь несколькими примерами. Основным языком для исследовательских работ сейчас является `Python`, лишь недавно принявший эстафету от `Matlab`, также популярного в сообществе.

Одним из основных признаков отличающих различные библиотеки является из подход к решению задачи построения дифференцируемого графа вычислений: Существует по сути 3 подхода:

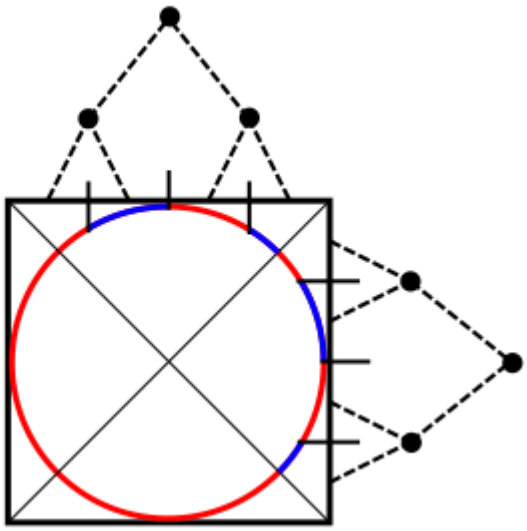


Рис. 1.3 – S-2 Проекция в двухмерном случае. Стороны квадрата рекурсивно разделяются и проектируются на окружность

–

– статический граф вычислений в котором архитектура не меняется в процессе работы (примеры — `Caffe`[15], `Keras`[16], `TensorFlow`[17] в `Python` и `Mocha`, `ArrayFire` в Julia)
– Динамический граф вычислений, использующий автоматическое дифференцирование. Примеры: `PyTorch`[18], `TensorFlow Eager`
– Смесь 1 и 2 подходов, позволяющая и задавать граф динамически и статически (`Apache MXNet`)

1.5 Анализ возможностей применения подхода transfer learning к проблеме геолокации с помощью глубокого обучения

Важным направлением в развитии сверточных нейронных сетей является передача обучения (transfer learning). Этот подход предполагает использование нейронных сетей, обученных на одних данных, для решения других типов задач. При этом применяется тонкая настройка сети и дообучение на данных от интересующей нас задачи. В результате сокращается время обучения и расширяется область применения предварительно обученных нейронных сетей.

1.6 Выводы

Подразумевается что разрабатываемая система действует автономно и не обладает дополнительной информацией о мире кроме информации о пространственной плотности объектов тестовой выборки.

1. Выполнен сравнительный анализ различных алгоритмов формирования признаков изображений (Локальные и Глобальные дескрипторы, Свёрточные нейронные сети) с точки зрения применимости к решению задачи визуальной локализации. Наиболее подходящей по признаку наилучших результатов признана система на основе свёрточной нейронной сети.
2. Был рассмотрен подход Transfer Learning и показана целесообразность его использования для решения поставленной задачи.
3. В качестве основной библиотеки для глубокого обучения была выбрана библиотека`pytorch` из-за наличия предобученных моделей, интуитивного подхода к реализации и простоты интеграции.

1.7 Постановка задачи УИР

Цель данного исследования — обзор современных алгоритмов глубокого обучения и применение их к задаче геолокации по серии изображений, а также создание ПО, демонстрирующего их работу на тестовых примерах.

Для достижения поставленной цели требуется

- изучить подробно применение современных алгоритмов глубокого обучения кк задачам Компьютерного Зрения,
- изучить современные нейросетевые методы работы с данными в виде последовательностей,
- изучить методы визуализации картографических данных,
- разработать способ генерации исходных данных для задачи визуальной локализации,
- спроектировать систему генерации признаков изображений для последующей классификации местоположений,
- разработать методы классификации вышеописанных векторов признаков,
- описать и разработать методы оценки точности классификации и провести сравнительный анализ различных подходов к задачам классификации и генерации признаков.

2. Разработка алгоритмов геолокации с помощью глубокого обучения

В этой главе описываются разработанные/модифицированные модели/методы/ алгоритмы, или/и описывается применение известных стандартных методов. Также, в конце главы обычно приводится общая архитектура программной системы, вытекающая из описанной теории. Приведенные ниже заголовки подразделов так же весьма примерные и сильно зависят от особенностей конкретной работы.

2.1 Формальная постановка задачи геолокации по серии изображений

Задача геолокации (визуальной локализации) в данном исследовании ставится как задача классификации изображений местности в соответствующие им области земной поверхности. Более формально, пусть:

- K — число ячеек в которых могли быть сделаны фотографии,
- N — число фотографий,
- m_i — число фотографий в i-том альбоме,
- Q — число альбомов,
- X — множество фотографий.
- Y — множество ячеек,
- $s2(\gamma)$ — сопоставление GPS координатам γ ячейке из Y. Для обучающей выборки известны значения $s2^{-1}$, то есть отображения $y \to \gamma$

ячейка — область земной поверхности, выделенная например с помощью S2 разбиения,

фотография — тензор $l \times q \times 3$ действительных чисел, l, q — ширина и высота соответственно[1],

альбом — упорядоченный набор из m_i фотографий объединённых по признаку нахождения в одной ячейке, K, N, Q определяются обучающей выборкой.

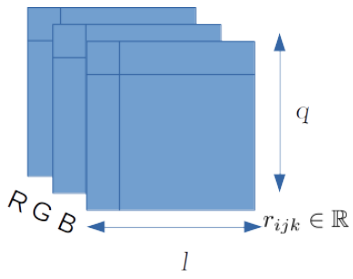


Рис. 2.1 – Фотография

Обучающая выборка — множество пар «фотография — ячейка», $\mathbf{X} = \{(X_j, y_j)|j = 1..N\}$, возможно объединённые в альбомы по m_j штук.

[1] индексы фотографии для простоты опущены

Требуется разработать:

– алгоритм, $\mathcal{F} : X \to Y$ который будет сопоставлять элементу x_j класс $\hat{y}_j$ наиболее близкий к тому которым x_j обладает в реальности, с набором параметров W_1
– алгоритм $\mathcal{G} : [X] \to Y$, где $[x]$ — список x, решающий эту задачу для альбома, $\mathcal{G}$ обладает набором параметров W_2

Обучение $\mathcal{F}$ и $\mathcal{G}$ состоит в подборе наборов параметров W_1 и W_2. Следует отметить что целесообразно использовать для реализации $\mathcal{G}$ параметры $\mathcal{F}$, а значит W_2 — суть W_1 и параметры, специфичные для обработки последовательностей. Оптимальные значения параметров W_1 и W_2 зависят от обучающей выборки.

Таким образом можно рассматривать $\mathcal{F}$ и $\mathcal{G}$ как отображение обучающей выборки в множество функций, отображающих в Y .

Следует также отметить что частота разбиения поверхности может быть различной, а значит для каждого масштаба возможно обучать различные версии $\mathcal{F}$ и $\mathcal{G}$.

Для простоты будем считать что перед подачей на вход алгоритмам изображения масштабируются и обрезаются если требуется. Процесс масштабирования будет описан ниже.

2.2 Выбор/разработка методов оценки точности работы алгоритмов геолокации

Для получения нижней оценки на точность алгоритма можно рассматривать в качестве ошибки предсказания расстояние от центра ячейки $(\hat{y})$ до реальных GPS координат фотографии $s2^{-1}(y)$, рассчитанное по поверхности земли $d(a, b)$.

Тогда ошибкой для выборки будет

$$E = \frac{1}{K} \sum_{i=1}^{K} d(y_j, C(\hat{y_j}))$$

Для обучения будем использовать скользящий контроль.

2.3 Модификация существующих решений в области для работы с серией изображений

Хотя разрабатываемая архитектура способна локализовать большое разнообразие изображений, многие изображения неоднозначны или не предоставляют достаточно информации, которая позволила бы их локализовать. Однако мы можем использовать тот факт, что фотографии естественным образом происходят в последовательности, например, альбомы, место съёмки которых часто значительно скоррелировано. Интуитивно, если мы с уверенностью можем локализовать некоторые из фотографий в альбоме, мы можем использовать эту информацию, чтобы локализо-

вать фотографии с неопределенным местоположением. Фотографии в альбоме — это последовательность, которая требует модели, которая запоминает состояние предыдущего примера, при рассмотрении текущего примера. Поэтому целесообразно использовать (LSTM) [19] для этой задачи. Теперь мы рассмотрим, как решить проблему прогнозирования геолокации последовательности фотографий с использованием LSTM.

2.3.1 Сети LSTM

Долгая краткосрочная память (Long short-term memory; LSTM) – особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. Они были представлены Зеппом Хохрайтер и Юргеном Шмидхубером (Jürgen Schmidhuber)[19] в 1997 году, а затем усовершенствованы и популярно изложены в работах многих других исследователей. Они прекрасно решают целый ряд разнообразных задач и в настоящее время широко используются.

LSTM разработаны специально, чтобы избежать проблемы долговременной зависимости. Поэтому запоминание информации на долгие периоды времени легко реализуется данной архитектурой.

Опишем основную операцию генератора паттренов на основе LSTM[20]: Блок LSTM имеет механизмы, позволяющие «запоминать» информацию для расширенного количества временных шагов. Мы используем блок LSTM со следующими преобразованиями, которые отображают входы для выходов по блокам в последовательных слоях и последовательных временных шагах:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

где $\circ$ - оператор поэлементного умножения, и две функции активации:

$$
\begin{aligned}
\sigma(\mathbf{x_i}) &= \frac{1}{1+e^{-\mathbf{x_i}}} \\
\tanh(\mathbf{x_i}) &= \frac{1-e^{-2\mathbf{x_i}}}{1+e^{-2\mathbf{x_i}}}
\end{aligned}
$$

Переменные:

– x_t — входной вектор,

– h_t — выходной вектор,

– c_t — вектор состояний,

– W , U и b — матрицы параметров и вектор,

– f_t, i_t и o_t — векторы вентилей,

– f_t — вектор вентиля забывания, вес запоминания старой информации,

– i_t — вектор входного вентиля, вес получения новой информации,

– o_t — вектор выходного вентиля, кандидат на выход.

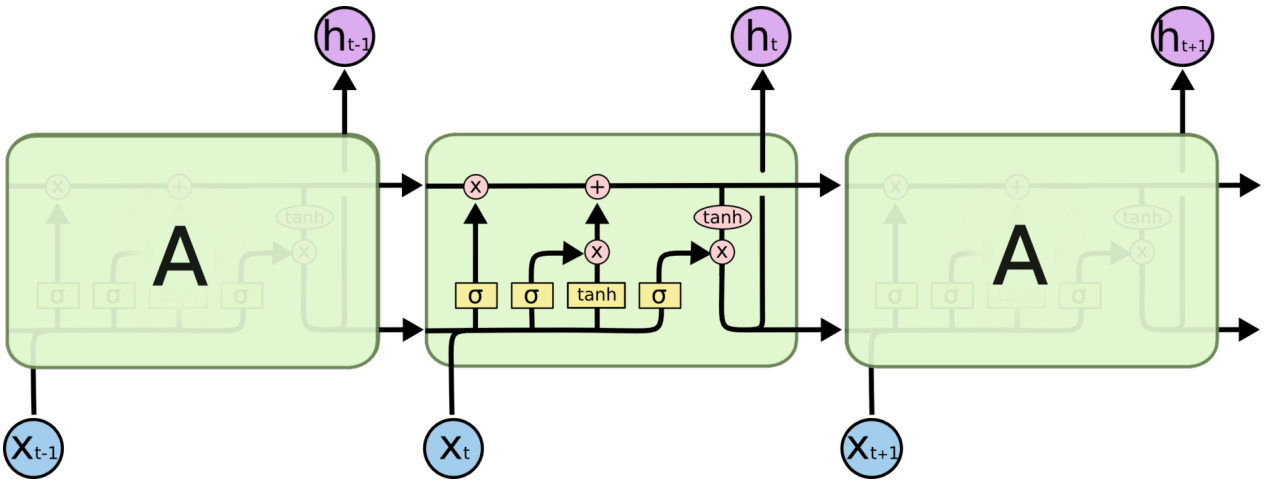


Рис. 2.2 – Повторяющийся модуль в LSTM сети

В приведенных выше преобразованиях ячейка памяти c_t хранит «долгосрочную» память в векторной форме. Другими словами, информация накапливается и кодируется до тех пор, пока временной интервал t не будет сохранен в c_t и будет передаваться только через один и тот же уровень на разных временных шагах. Учитывая входы c_t и h_t, вход и “забывающий” элемент f_t помогут ячейке памяти решить, как перезаписать или сохранить информацию в памяти. Выходной шлюз далее позволяет блоку LSTM решить, как получить информацию о памяти для генерации текущего состояния h_t, которое передается как на следующий уровень текущего временного шага, так и на следующий шаг времени текущего уровня. Такие решения принимаются с использованием параметров скрытого слоя W и b с разными индексами: эти параметры будут выведены на этапе обучения сети.

2.4 Разработка метода геолокации по серии изображений используя выбранные/разработанные выше алгоритмы/методы

2.4.1 Выбор типа свёрточной сети для классификации

Существует множество исследований, сравнивающих производительность различных архитектур свёрточных сетей [21].

AlexNet

В 2012 году на конкурсе ILSVRC по классификации изображений впервые победила нейронная сеть — AlexNet, достигнув top-5 ошибки 15,31%. Для сравнения, метод, не использующий свёрточные нейронные сети, получил ошибку 26,1%. В AlexNet были собраны новейшие на тот момент техники для улучшения работы сети. Обучение AlexNet из-за количества параметров сети происходило на двух GPU, что позволило сократить время обучения в сравнении с обучением на CPU. Также оказалось, что использование функции активации ReLU (Rectified Linear Unit) вместо более традиционных функций сигмоиды и гиперболического тангенса позволило снизить количество эпох обучения в шесть раз. Формула ReLU следующая:

$$y = max(0, x)$$

ReLU позволяет побороть проблему затухания градиентов, свойственную другим функциям активации.Помимо прочего, в AlexNet была применена техника отсева (Dropout). Она заключается в случайном отключении каждого нейрона на заданном слое с вероятностью p на каждой эпохе. После обучения сети, на стадии распознавания, веса слоёв, к которым был применён dropout [22], должны быть умножены на $1/p$. Dropout — по-сути регуляризатор, препятствующий переобучению. Для объяснения эффективности данной техники существует несколько интерпретаций. Первая заключается в том, что dropout заставляет нейроны не полагаться на соседние нейроны, а обучаться распознавать более стойкие признаки. Вторая, более поздняя, состоит в том, что, обучение сети с dropout представляет собой аппроксимацию обучения ансамбля сетей, каждая из которых представляет сеть без некоторых нейронов Таким образом, конечное решение принимает не одна сеть, а ансамбль, каждая сеть которого обучена по-разному, тем самым снижается вероятность ошибки.

ResNet

Победителем ILSVRC 2015 с top-5 ошибкой в 3,57 % стал ансамбль из шести сетей типа ResNet (Residual Network), разработанный в Microsoft Research . Авторы ResNet заметили, что с повышением числа слоёв свёрточная нейронная сеть может начать деградировать — у неё понижается точность на валидационном множестве. Так как падает точность и на тренировочном множестве, можно сделать вывод, что проблема состоит не в переобучении сети. Было сделано предположение, что если свёрточная нейронная сеть достигла своего предела точности на некотором слое, то все следующие слои должны будут выродиться в тождественное преобразование, но из-за сложности обучения глубоких сетей этого не происходит. Для того чтобы «помочь» сети, было пред-

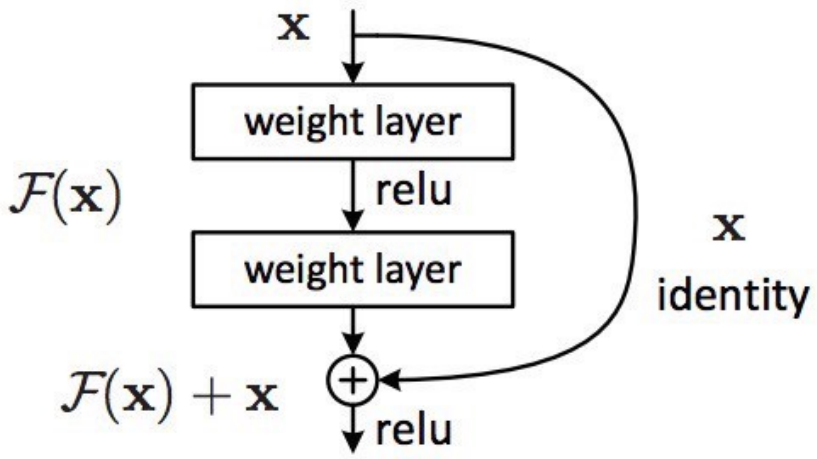
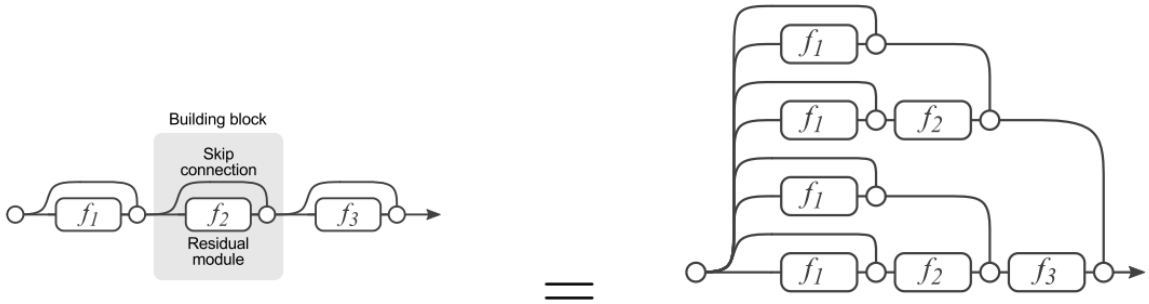


Рис. 2.3 – Пропускающее соединение

ложено ввести пропускающие соединения (Shortcut Connections), изображённые на рисунке 2.3.

Пускай оригинальная сеть должна вычислять функцию $H(X)$. Определим её остаточную функцию как $\mathcal{F}(X) = H(x) - x$, которая, в теории, будет проще обучить. Добавив пропускающие соединения, сеть учится остаточной функции, которая затем складывается с тождественным преобразованием. Анализ[23] в показал, что глубокие остаточные нейронные сети можно считать ансамблем, состоящим из более мелких остаточных нейронных сетей,чья эффективная глубина увеличивается в процессе обучения.



(a) Conventional 3-block residual network (b) Unraveled view of (a)

Рис. 2.4 – Идея для утверждения об ансамбле

SqueezeNet

Позволяет получить точность AlexNet при уменьшении числа параметров в 50 раз. Основные идеи:

– Свёртки 3×3заменены на свёртки 1×1. Каждая такая замена в 9 раз уменьшает число параметров

– На вход оставшихся свёрток 3×3 пробуют подавать только маленькое число каналов

– Уменьшение размера делается как можно позднее, чтобы свёрточные слои имели большую площадь активации.

Эти три стратегии привели к тому, что авторы создали «fire module», рисунок 2.5

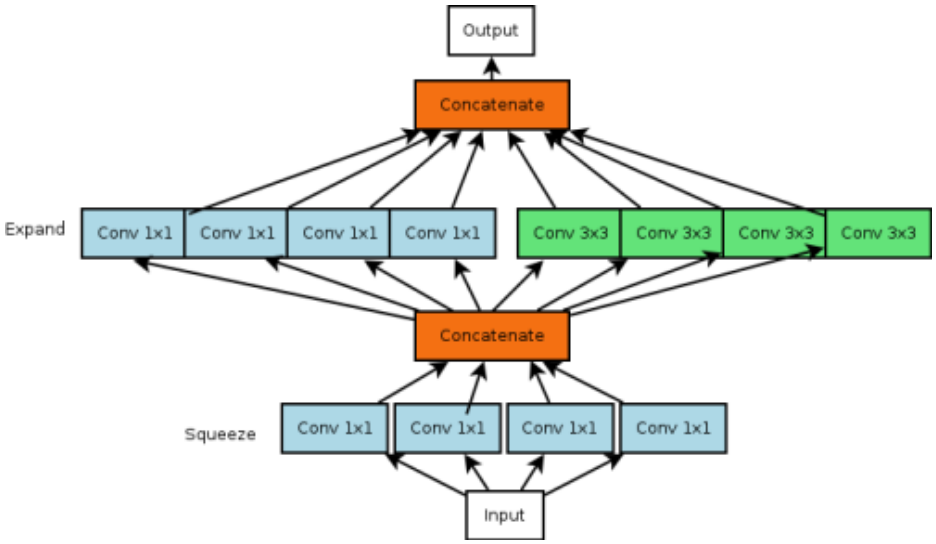


Рис. 2.5 – FireModule

Из таких модулей и собирается вся сеть. Кроме того, авторы применили несколько классических подходов:

Отказ от полносвязного слоя на последнем уровне. В принципе, это достаточно модно и неплохо влияет на производительность. Для того чтобы это достичь авторы берут выход последнего свёрточного слоя, разделяют его на N частей, и усредняют каждую часть через avg-pooling по числу каналов. Эти выходы пулинга подаются на обучающие нейроны. В некоторых из версий сети авторы используют элементы сетей с пропускающими (residual) соединениями. Это увеличивает точность.

На рисунке 2.6:
квадраты — обучение сети заново,
шестиугольники — дообучение всей сети,
круги — переобучение только последнего слоя.

2.5 Показатели точности классификации

В качестве показателей точности используются:

– Чувствительность (SEN) и специфичность (SPE)

– $F1$ мера

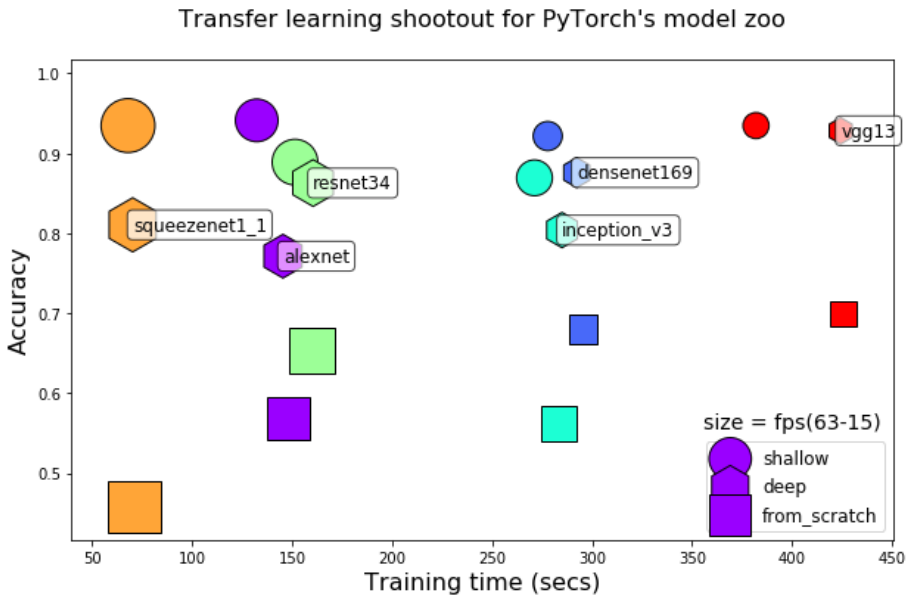


Рис. 2.6 – Сравнение различных архитектур свёрточных сетей и различных способов обучения

Опишем формулы вычисления этих показателей:

$$SEN = TP/P = TP/(TP + FN)$$

$$SPE = TN/N = TN/(TN + FP)$$

$$PRC = TP/(TP + FP)$$

$$F_1 = 2 \cdot \frac{PRC \cdot SEN}{PRC + SEN}$$

Пусть есть два класса результатов: положительный (positive) и отрицательный (negative). Тогда на выходе классификатора может наблюдаться четыре различных ситуации:

Если результат классификации положительный, и истинное значение тоже положительное, то речь идет об истинно-положительном значении (true-positive, TP) Если результат классификации положительный, но истинное значение отрицательное, то речь идет о ложно-положительном значении (false-positive, FP) Если результат классификации отрицательный, и истинное значение тоже отрицательное, то речь идет об истинно-отрицательном значении (true-negative, TN) Если результат классификации отрицательный, но истинное значение положительно, то речь идет о ложно-отрицательном значении (false-negative, FN)

Функции потерь

В задачах классификации наиболее естественным выбором является пороговая функция потерь $\mathcal{L}(y, y') = [y' \neq y]$. Когда функция потерь разрывна, минимизация эмпирического риска оказывается сложной задачей комбинаторной оптимизации. Во многих практически важных слу-

чаях эта сводится к поиску максимальной совместной подсистемы в системе неравенств (число неравенств совпадает с число объектов обучения m) и является NP-полной.

Наряду с пороговыми фукциями потерь используются всевозможные их непрерывные аппроксимации, что позволяет применять достаточно эффективные классические методы непрерывной оптимизации, в том числе градиентные методы. Более того, оказывается, что использование некоторых аппроксимаций способно улучшать обобщающую способность алгоритма классификации.

Интересным примером сложной функции потерь является перекрёстная энтропия.

В теории информации перекрёстная энтропия между двумя распределениями вероятностей измеряет среднее число бит, необходимых для опознания события из набора возможностей, если используемая схема кодирования базируется на заданном распределении вероятностей q, вместо «истинного» распределения p.

Перекрестная энтропия для двух распределений p и q над одним и тем же вероятностным пространством определяется следующим образом:

$$\mathrm{H}(p,q) \stackrel{\mathrm{df}}{=} \mathrm{E}_p[-\log q] = \mathrm{H}(p) + D_{\mathrm{KL}}(p\|q),$$

где $H(p)$— энтропия p, и $D_{\mathrm{KL}}(p||q)$ — расстояние Кульбака—Лейблера отp до q (также известная как относительная энтропия).

Для дискретногоp иq это означает

$\mathrm{H}(p,q) = -\sum_x p(x)\ \log q(x).$

Рассматриваем задачу классификации с C классами.

Вычислительно функция Потерь может быть описана как:

$$\mathrm{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

или для взвешенной функции потерь:

$$\mathrm{loss}(x, class) = weight[class]\left(-x[class] + \log\left(\sum_j \exp(x[j])\right)\right)$$

Потеря усредняются по наблюдениям для каждой подвыборки.

2.6 Архитектура Модели для Альбомов

Основная структура модели для последовательности такова: учитывая изображение, мы извлекаем вектор признаков изображения (до слоя SoftMax) в PlaNet. Этот вектор подается в блок LSTM. Выходной вектор LSTM затем подаётся в слой SoftMax, который выполняет классификацию в ячейки S2. Мы подаём альбом в модель в хронологическом порядке. В генерации признаков

мы переиспользуем параметры модели для одиночного изображения. Во время обучения мы сохраняем их фиксированными и только обучать параметры LSTM и слой SoftMax.

Проблема с этой простой моделью LSTM заключается в том, что многие альбомы содержат вначале несколько изображений, которые не содержат полезной визуальной информации. Благодаря своей однонаправленности эта модель не может исправить неверные прогнозы, что происходят в начале последовательности после наблюдения фото с уверенным местоположением. По этой причине мы сейчас оценить модель, в которой LSTM использует множество фотографий от альбома, прежде чем сделать свое первое предсказание. Смещение метки. Идея этой модели состоит в том, чтобы сдвинуть поиск поисков, что вывод отложен на несколько временных шагов. Основная мотивация этой идеи состоит в том, что модель может накапливать информацию из нескольких изображений в последовательность перед предсказаниями. Тем не менее, мы обнаружили, что использование смещений не улучшает локализацию. Предположим, что из-за ввода входного изображения на выходные метки становится более сложным, затрудняя прогнозирование для всех фотографий, улучшая прогнозы только для конечности количество фотографий. Более того, этот подход не решает проблему повсеместно: например, если мы компенсируем метка на 2 шага, но первое изображение Фертильность возникает только после 3 шагов, прогноз для первого изображение, вероятно, все еще будет неправильным. Чтобы исправить это, теперь мы сидером, которые определяют их прогнозы на всех изображениях в последовательности, а не только предыдущие. Повторяющиеся последовательности. Сначала мы оцениваем модель, которая обученный по последовательностям включая два экземпляра одной и той же последовательности. для эта модель, мы принимаем только прогнозы для изображений из вторая половина последовательности (т. е. повторяющаяся часть).

При времени вывода, передавая последовательность в модель в первый раз можно рассматривать как стадию кодирования, где LSTM создает внутреннее состояние на основе изображений. Второй проход представляет собой этап декодирования, где на каждом изображении, LSTM делает прогноз на основе его состояния, аренда изображение. Результаты показывают, что этот подход превосходит однопроходные LSTM, достигающие 7,8% относительное улучшение на уровне улицы, ценой двоякого увеличение времени вывода. Однако, В результате мы обнаружили проблему с этим подходом: если в начале последовательности, они, как правило, привязаны к последнему катиона в последовательности, поскольку модель учится полагаться на его предыдущее предсказание. Поэтому прогнозы с конца последовательности переносятся в начало. Двунаправленный LSTM. Известная нейронная сеть ar- архитектуры, которая обусловливает прогнозы в целом последовательности являются двунаправленными LSTM (BLSTM). Эта модель можно рассматривать как конкатенацию двух моделей LSTM, где первая выполняет передний проход, а вторая вы-

полняет обратный проход по последовательности. Двунаправленные LSTM не могут быть обучены с помощью обратного распространения во времени с усечением и, следовательно, для LSTM до полной длины последовательности. Чтобы уменьшить вычислительную стоимость обучения, нам пришлось ограничить продолжительность последовательности до 25 изображений. Это приводит к снижению общей точности, поскольку более длинные альбомы обычно дают большую точность чем короткие. Поскольку наши эксперименты по этим данным не сопоставимы с предыдущими, мы также оценили повторяющуюся модель LSTM на усеченных последовательностях до 25 изображений. Как показывают результаты BLSTM явно превосходят повторные LSTM (Относительное улучшение на уровне улицы на 16,6%). Однако, потому что они не подходят для длинных последовательностей, повторяющиеся модель может быть предпочтительнее на практике.

2.7 Выводы

Решено было использовать архитектуру resnet34 компромисс между временем обучения и качеством. В рамках исследования архитектура сети `SquezeNet` оказалась малоприменимой к данной задаче, несмотря на свои особенности (малое количество параметров, лёгкость обучения).

Задача сведена к задаче классификации изображений для решения которой могут быть применены стандартные средства, а именно свёрточные сети современной архитектуры.

Необходимо перечислить, какие теоретические результаты были получены с указанием степени новизны. Например: «Была разработана такая-то модель. Она представляет собой адаптированную версию модели X, в которой уравнение Z заменено на уравнение Z'». Еще пример: «Была предложена такая-то архитектура, она отличается от типовой в том-то и том-то. Это позволяет избежать таких-то проблем.». При этом не следует заниматься «высасыванием из пальца»: «Поставленная задача является типовой; для ее решения применены стандартные средства (перечислить, какие).».

3. Инженерная часть: Проектирование и Реализация ПО

Для простоты считаем что решаемая задача заключается в классификации изображений на 10 классов (по аналогии с задачей CIFAR-10 по отношению к ImageNet) В связи с этим многие элементы системы могут быть упрощены: В частности нет необходимости описывать подробно работу с пространственным разбиением.

3.1 Проектирование системы геолокации по серии изображений

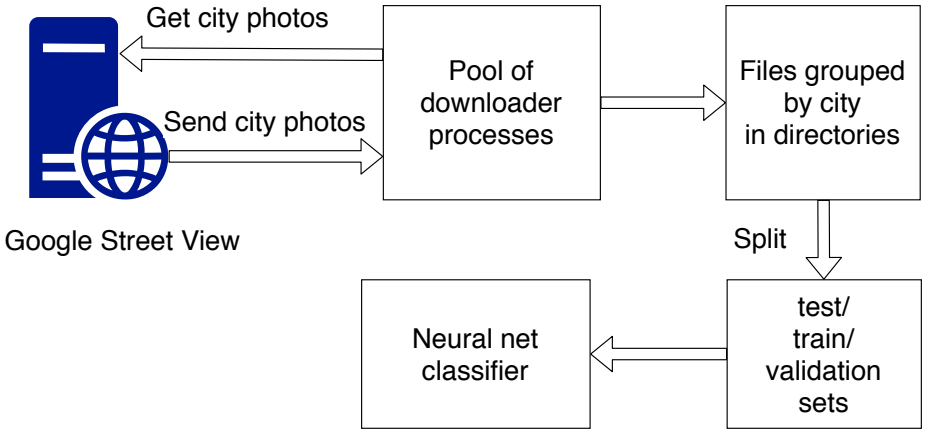


Рис. 3.1 – Схема для генерации датасета

Опишем на высоком уровне работу системы: Мы собираем датасет из панорам `Google Street View` в 10 городах, определённых своими координатами. Запросы в `API` выполняются параллельно пулом из 4 процессов, которые раскладывают изображения в директории соответствующие классам. Эти 10 классов можно заменить K `S2`- ячейками и набирать из каждой ячейки фотографии с другого сервиса, такого как Flickr (см [1]). Для 10 классов можно набрать по 100 представителей каждого класса.

После этого мы разбиваем этот набор в соотношении $8, 1, 1$ обучающая, тестовая, валидационная выборки, после чего приводим структуру файлов к формату, который понимает функция `torchvision.datasets.ImageFolder`, а именно: кладём фотографии в директории, соответствующие фазе обработки (train, test, val).

Этот процесс показан на рисунке 3.1

Так как мы используем нейросетевые методы имеет смысл использовать модель классификации end-to-end, то есть: $y = CLS(x)$, где x — фотография, нормированная для улучшения процесса классификации, y — класс, относимый сетью этой фотографии.

Для реализации этой модели мы воспользуемся предобученными моделями из программного пакета `torchvision`, также предоставляющего нормирующие преобразования, функции аугментации данных (для расширения обучающей выборки), потокового чтения данных.

3.2 Архитектура подсистемы классификации

Из доступных предобученных моделей выберем несколько подходящих под следующие критерии:

- параметры полностью помещаются в память GPU автора (4GB), что необходимо для обучения/дообучения;
- архитектура позволяет дообучать модель;
- сеть демонстрирует точность выше 0.8 на датасете ImageNet.

Таким критериям отвечают: resNet34, resNet50, squeezeNet, Inception_V3, DenseNet.

Все эти сети являются глубокими свёрточными нейросетями, а потому подробное рассмотрение внутреннего устройства и теоретических преимуществ функционирования помимо описанных в Разделе 2 выходит за рамки данной работы.

Благодаря модульности и динамичности пакета `pytorch`, обучающий алгоритм и алгоритм визуализации не зависят от используемой архитектуры сети.

Оценивать точность будем при помощи `sklearn.metrics` содержащего реализации помимо прочего процедур вычисления $F1$-меры, матрицы неточностей, отчёта о классификации; и пакета `torch.nn`, реализующего функцию потерь перекрёстной энтропии и множество нейросетевых примитивов, таких как: полносвязные и свёрточные слои, SoftMax, Pooling и различные нелинейности.

3.3 Взаидодействие с моделью

Библиотека предоставляет возможность сериализации модели что позволяет переисользовать исследовательский прототип в приложениях. Так можно обучить модель и затем встроить её в более крупную систему, такую как веб или мобильное приложение.

Взаимодействие с исследовательским прототипом осуществляется с помощью механизмов интерактивных рабочих тетрадей `Jupyter Notebook`.

3.4 Выводы

Были разработаны модули генерации датасета, разбиения на обучающую/тестовую выборки и классификации. Были реализованы процедуры для обучения и визуализации работы сети, а также механизмы оценки точности работы сети.

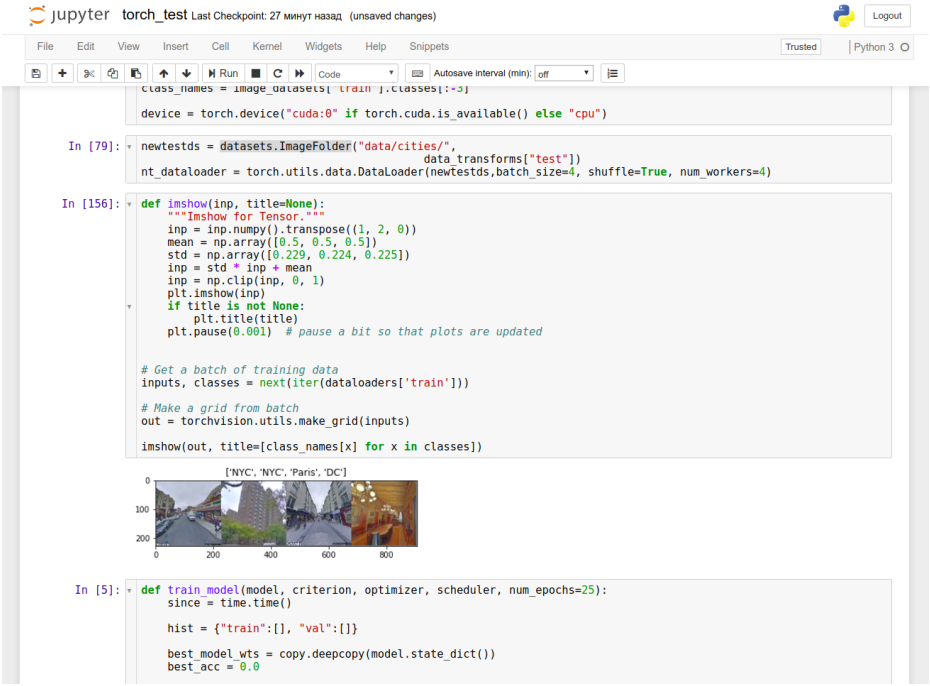


Рис. 3.2 – интерактиная рабочая тетрадь jupyter

4. Экспериментальная проверка Алгоритмов геолокации

4.1 Состав и структура реализованного программного обеспечения

4.2 Описание обучающих и тестовых данных

Для тестирования разработанной системы использовались снимки `Google Street View` из 10 городов снятые в панорамном режиме с разрешением 224×224. Код, генерирующий эти снимки представлен на листинге А.3.

Так как `API Google Street View` отвечает картинкой и не для всех координат в городе существуют панорамы, отфильтровывать ошибки можно просто используя размер картинки как хэш функцию. Коллизии маловероятны из-за того что обычно панорамы на порядок отличаются по размеру от картинки, обозначающей ошибку. Интересующая нас часть листинга представлена ниже.

Снимки генерируются методом случайного сдвига точки к которой делается запрос и случайного поворота.

Для разделения на тестовую и обучающую выборки также используются скрипты.

4.3 Результаты экспериментов

4.3.1 Эксперимент с 1 изображением

Было проведено обучение модели на основе свёрточной сети `resnet34` с использованием подхода `transfer learning`. Для оптимизации использовался метод Стохастического градиентного спуска с параметрами скорость обучения $= 0,001$momentum $= 0.9$. В процессе обучения каждые 7 шагов скорость обучения сокращалась в 10 раз. Гиперпараметры подобраны методом поиска по решетке со скользящим контролем. Для этого использовался метод `sklearn.model_selecti`

В качестве функции потерь использовалась «Перекрёстная энтропия». На листинге ниже пред-

```
# check if the downloaded image was invalid and if so remove it
...
if os.path.isfile(filepath):
  size = os.path.getsize(filepath)
  if size == FAILED_DOWNLOAD_IMAGE_SIZE:
    os.remove(filepath)
    misses += 1
  else:
    num_imgs += 1
...
```

Выборка	Точность распознавания
Обучение	0.711
Валидация	0.424
Тест	0.602

Таблица 4.1 – Точность распознавания для архитектуры `resnet34` с переобучением всех слоёв

Выборка	Точность распознавания
Обучение	0.421
Тест	0.429
Валидация	0.384

Таблица 4.2 – Точность распознавания для архитектуры `resnet34` с переобучением последнего слоя

ставлены определения для обучения сети. Процедура обучения сети описана в листинге А.2

```
model_ft = models.resnet34(pretrained=True)
num_ftrs = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_ftrs, len(class_names))

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that the last layer parameters are being optimized
optimizer_ft = optim.SGD(model_ft.fc.parameters(), lr=0.001, momentum=0.9

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0
```

Для варианта с переобучением последнего слоя:

Обучение проходило 40 эпох. Время обучения — ≈ 5 мин.

`F1`-взвешенная оценка для классификации 0.5101 Результаты классификации представлены на иллюстрациях и в таблице 4.1.

Для варианта с переобучением всей сети:

Обучение проходило 30 эпох. Время обучения — ≈ 9 мин.

`F1`-взвешенная оценка для классификации 0.50 Результаты классификации представлены на иллюстрациях и в таблице 4.2.

В сравнении с другими архитектурами resNet34 выиграл по времени обучения у resNet50, Inception_V3, DenseNet и по качеству классификации у squeezeNet.

4.3.2 Эксперимент с альбомом

Мы сравниваем эту модель с моделью и базовый уровень, который просто усредняет прогнозы по одному изображению всех изображений в альбоме и назначает среднее для всех изображений. Усреднение в альбомах ($\mathcal{G} = \mathbb{E}[\mathcal{F}(X)]$) уже дают значительное улучшение по сравнению с одним изображением (45,7% на уровне улиц), поскольку у него больше вмятины предсказания к неоднозначным изображениям. Однако, LSTM модель явно превосходит технику усреднения (50,5%

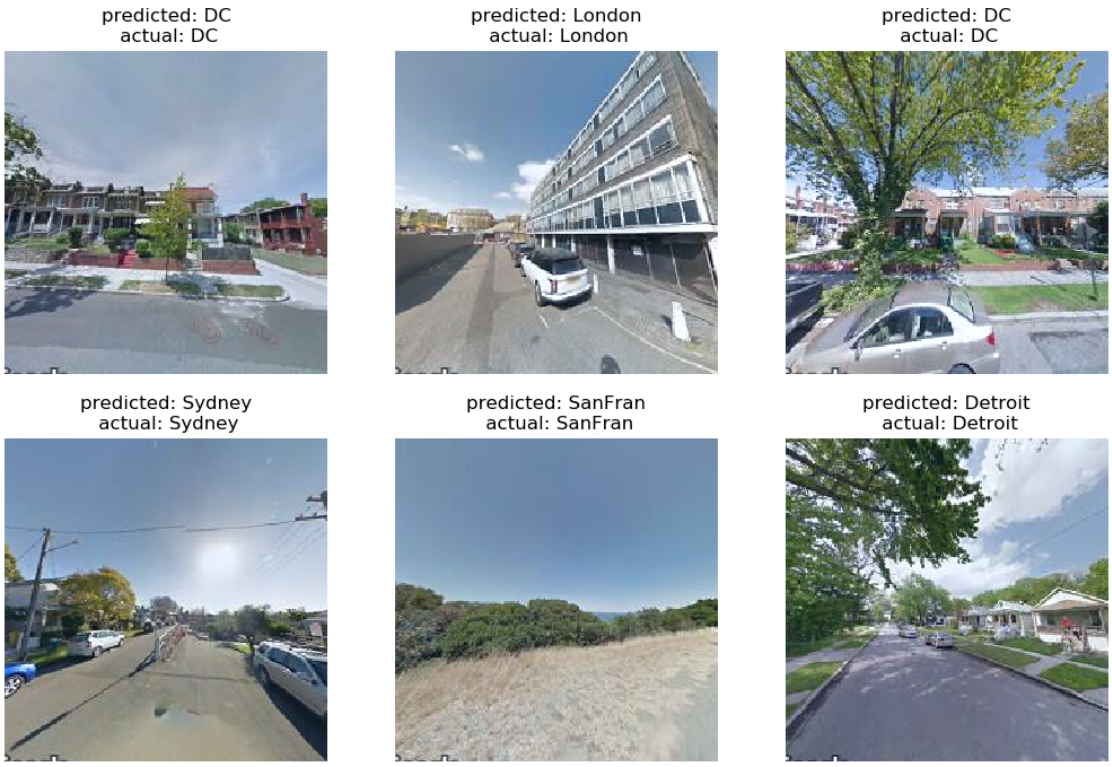


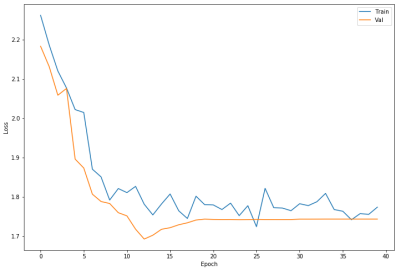
Рис. 4.1 – Результат обучения модели с resNet18 по 1 изображению.

Пример удачно выпавших классов

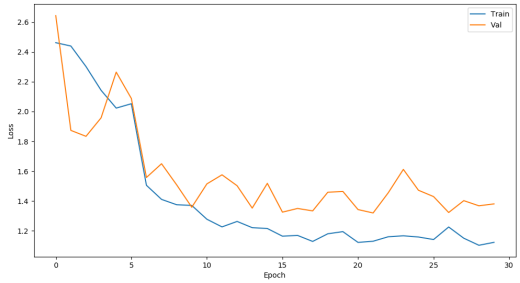
относительное улучшение уровня улицы). Визуальный контроль результатов показали, что за достоверностью следуют несколько изображений с более низким местоположением уверенность, модель LSTM присваивает низкий уровень доверия изображения, расположенные вблизи изображения с высокой степенью достоверности в то время как $\mathcal{F}$ может «прыгать» (менять предположение о расположении альбома), модель LSTM имеет тенденцию прогнозировать близкие местоположения, кроме случаев когда есть убедительные доказательства изменения местоположения. LSTM модель работает лучше усреднения из-за того что оно присваивает всем изображениям в альбоме одинаковые уровни значимости и не может производить точные прогнозы для альбомов которые включают разные местоположения (например, альбомы поездок).

4.4 Сравнение реализованного программного обеспечения с существующими аналогами

В качестве аналогов выступают системы PlaNet и Im2GPS. Для сравнения будем использовать оценку точности на уровне региона (750 км), так как континентов ≈ 6 а регионов несколько больше.



(а) для варианта с переобучением последнего слоя



(б) при дообучении всех слоёв сети.

Рис. 4.2 – Значение функции потерь на Обучающей и Валидационной выборках по эпохам

Система	Точность
PlaNet	53.6%
IM2GPS	35.4%
Разрабатываемая система	**44.7%**

Таблица 4.3 – Сравнение показателей точности

4.5 Выводы

Были разработаны скрипты для генерации «датасета», разбиения на обучающую / тестовую / валидационную, приведения датасета к читаемому виду. Также были исследованы методы `transfer learning`, был произведён их сравнительный анализ и была выявлена следующая закономерность: дообучение последнего слоя не уступает по качеству переобучению всей сети, несмотря на более высокие показатели качества на тестовой выборке.

Было проведено сравнение с более ранними разработками и достигнут уровень точности сопоставимый с ними.

Заключение

– были проанализированы методы геолокации по изображениям, исторические решения задачи визуальной локализации, алгоритмы разбиения земной поверхности, существубщих библиотек для глубокого обучения, подход переноса параметров.

  В итоге были выбраны библиотеки и модели/алгоритмы для дальнейшего изучения.
– были описаны использованные и модифицированные алгоритмы классификации, архитектуры нейросетей, способы оценки точности классификации, функции потерь;
– был спроектирована система для геолокации серий изображений;
– были проведены испытания разработанного прототипа в рамках подзадачи общей задачи геолокации.

Была использована библиотека `pytorch` для упрощения работы с различными архитектурами.

Было выяснено что для задачи многклассовой классификации дообучение всей сети приводит лишь к переобучению к тестовой выборке, так как результат на валидационной выборке и $F1$-мера для случая переобучения последнего слоя близки к случаю дообучения всех параметров сети.

На модельном датасете получены результаты, сопоставимые с SOTA в данной области:
$F_1 = 0.452$
Точность $= 0,447$
значительно превосходящие вероятность случайного угадывания.

Направления будущей работы:

– В рамках практимки улучшить точность классификации и увеличить её охват
– Описать более подробно работу с последовательностями
– Рассмотреть возможность применения подобной архитектуры к другим задачам машинного обучения/ компьютерного зрения
– Визуализировать состояние обученных слоёв сети.

Список литературы

1. Hays James, Efros Alexei A. IM2GPS: estimating geographic information from a single image // Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on / IEEE. — 2008. — P. 1–8.

2. Geometric reasoning under uncertainty for map-based localization / William B Thompson, Carolyn M Valiquette, Bonnie H Bennett, Karen T Sutherland // Spatial Cognition and Computation. — 1999. — Vol. 1, no. 3. — P. 291–321.

3. Weyand Tobias, Kostrikov Ilya, Philbin James. Planet-photo geolocation with convolutional neural networks // European Conference on Computer Vision / Springer. — 2016. — P. 37–55.

4. Lin Tsung-Yi, Belongie Serge, Hays James. Cross-view image geolocalization // Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on / IEEE. — 2013. — P. 891–898.

5. Retrieving landmark and non-landmark images from community photo collections / Yannis Avrithis, Yannis Kalantidis, Giorgos Tolias, Evaggelos Spyrou // Proceedings of the 18th ACM international conference on Multimedia / ACM. — 2010. — P. 153–162.

6. Skyline2gps: Localization in urban canyons using omni-skylines / Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, Matthew Brand // Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on / IEEE. — 2010. — P. 3816–3823.

7. Color histogram. — 2018. — Jan. — Page Version ID: 821304796. online; accessed: `https://en.wikipedia.org/w/index.php?title=Color_histogram&oldid=821304796` (online; accessed: 2018-04-16).

8. Stricker Markus, Dimai Alexander. Spectral covariance and fuzzy regions for image indexing // Machine vision and applications. — 1997. — Vol. 10, no. 2. — P. 66–73.

9. Haralick Robert M, Shanmugam Karthikeyan et al. Textural features for image classification // IEEE Transactions on systems, man, and cybernetics. — 1973. — no. 6. — P. 610–621.

10. Tamura Hideyuki, Mori Shunji, Yamawaki Takashi. Textural features corresponding to visual perception // IEEE Transactions on Systems, man, and cybernetics. — 1978. — Vol. 8, no. 6. — P. 460–473.

11. Lowe David G. Object recognition from local scale-invariant features // Computer vision, 1999. The proceedings of the seventh IEEE international conference on / Ieee. — Vol. 2. — 1999. — P. 1150–1157.

12. Bay Herbert, Tuytelaars Tinne, Van Gool Luc. Surf: Speeded up robust features // European conference on computer vision / Springer. — 2006. — P. 404–417.

13. Ke Yan, Sukthankar Rahul. PCA-SIFT: A more distinctive representation for local image descriptors // Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on / IEEE. — Vol. 2. — 2004. — P. II–II.

14. Shechtman Eli, Irani Michal. Matching local self-similarities across images and videos // Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on / IEEE. — 2007. — P. 1–8.

15. Caffe: Convolutional Architecture for Fast Feature Embedding / Yangqing Jia, Evan Shelhamer, Jeff Donahue et al. // arXiv preprint arXiv:1408.5093. — 2014.

16. Chollet François et al. Keras. — `https://keras.io`. — 2015.

17. Abadi Martín, Agarwal Ashish, Barham Paul et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. — 2015. — Software available from tensorflow.org. Access mode: `https://www.tensorflow.org/`.

18. Automatic differentiation in PyTorch / Adam Paszke, Sam Gross, Soumith Chintala et al. // NIPS-W. — 2017.

19. Hochreiter Sepp, Schmidhuber Jürgen. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.

20. Gers Felix A, Schmidhuber Jürgen, Cummins Fred. Learning to forget: Continual prediction with LSTM. — 1999.

21. Сикорский ОС. Обзор свёрточных нейронных сетей для задачи классификации изображений // Новые информационные технологии в автоматизированных системах. — 2017. — no. 20.

22. Dropout: A simple way to prevent neural networks from overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky et al. // The Journal of Machine Learning Research. — 2014. — Vol. 15, no. 1. — P. 1929–1958.

23. Veit Andreas, Wilber Michael J, Belongie Serge. Residual networks behave like ensembles of relatively shallow networks // Advances in Neural Information Processing Systems. — 2016. — P. 550–558.

A. Фрагменты исходных текстов программ

```
def visualize_model(model, num_images=6):
# bookkeeping and setup of figues
    was_training = model.training # keep the previous state of model

   model.eval() # set eval mode (BatchNorm and Dropout work differently)
    images_so_far = 0
    fig = plt.figure()

    with torch.no_grad():
        for i, (inputs, labels) in enumerate(dataloaders['val']):
 # Move inputs and labels to GPU
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
     # Draw Images and predictions
                images_so_far += 1
                ax = plt.subplot(num_images//3, 3, images_so_far)
                ax.axis('off')
                ax.set_title(
                  'predicted: {}\n actual: {}'.
                  format(class_names[preds[j]],
                         class_names[labels[j]]))
                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

Листинг А.1 – Листинг процедуры визуализации

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()

    hist = {"train":[], "val":[]}

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                scheduler.step()
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0

            # Iterate over data.
```

```
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                # zero the parameter gradients
                optimizer.zero_grad()

                # forward
                # track history if only in train
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)

                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                # statistics
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc =
            running_corrects.double() / dataset_sizes[phase]

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                phase, epoch_loss, epoch_acc))
            hist[phase] += [[epoch_loss, epoch_acc]]

            # deep copy the model
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())

        print()

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
    model.load_state_dict(best_model_wts)
    return model, hist
```

Листинг А.2 – Листинг процедуры обучения

```
import math
import os
import random
import sys
from multiprocessing.pool import ThreadPool
import urllib

import file_utils

cities = {
"Paris" : (48.8567,2.3508),
"London" : (51.5072,-0.1275),
"Barcelona" : (41.3833,2.1833),
"Moscow" : (55.7500,37.6167),
"Sydney" : (-33.8650,151.2094),
"Rio" : (-22.9068,-43.1729),
"NYC" : (40.7127,-74.0059),
```

```
  "SanFran" : (37.7833,-122.4167),
  "Detroit" : (42.3314,-83.0458),
  "DC" : (38.9047,-77.0164)
}

# radius of the Earth
R = 6378.1

# radius of images around center of city
IMAGE_RADIUS = 10

# number of images to download from each city
NUM_IMAGES_PER_CITY = 200

# size of failed-download image
FAILED_DOWNLOAD_IMAGE_SIZE = 3464

# place key in a file in the Geo-Localization directory
# as the only text in the file on one line
KEY_FILEPATH = "../api_key.key"
API_KEY = file_utils.load_key(KEY_FILEPATH)
GOOGLE_URL = ("http://maps.googleapis.com/maps/api/streetview?"
              "size=256x256&fov=120&pitch=10&key=" + API_KEY)
IMAGES_DIR = '../data/cities/'


def download_images_for_city(city, lat, lon):
    print('downloading images of {}'.format(city))
    num_imgs = 0
    misses = 0
    cur_directory = os.path.join(IMAGES_DIR, city)
    if not os.path.exists(cur_directory):
        os.makedirs(cur_directory)

    while num_imgs < 100:

        # randomly select latitude and longitude in the city
# bearing is 90 degrees converted to radians.
        brng = math.radians(random.uniform(0, 360))
        d = random.uniform(0, IMAGE_RADIUS)
# current lat point converted to radians
        lat_rad = math.radians(lat)
# current long point converted to radians
        lon_rad = math.radians(lon)
        rand_lat = math.asin(math.sin(lat_rad)*math.cos(d/R) +
    math.cos(lat_rad)*math.sin(d/R)*math.cos(brng))
        rand_lon = lon_rad + math.atan2(
    math.sin(brng)*math.sin(d/R)*math.cos(lat_rad),
    math.cos(d/R)-math.sin(lat_rad)*math.sin(rand_lat))
        rand_lat = math.degrees(rand_lat)
        rand_lon = math.degrees(rand_lon)

        # download image
        filename = 'lat-{}-lon-{}.jpg'.
    format(round(rand_lat, 4), round(rand_lon, 4))
        filepath = os.path.join(cur_directory, filename)
        url = GOOGLE_URL +
     "&location=" +
    str(rand_lat) + "," +
    str(rand_lon) + "&heading=" + str(brng)
        res = urllib.request.urlretrieve(url, filepath)

# check if the downloaded image was invalid and if so remove it
        if os.path.isfile(filepath):
            size = os.path.getsize(filepath)
```

```
            if size == FAILED_DOWNLOAD_IMAGE_SIZE:
                os.remove(filepath)
                misses += 1
            else:
                num_imgs += 1

    print('invalid photo of {} downloaded {} times'.format(city, misses))
    file_utils.upload_directory_to_aws(cur_directory)

def download_images():

    # download images for each city in a different thread
    num_threads = 8
    pool = ThreadPool(num_threads)
    for city, (lat, lon) in cities.items():
        pool.apply_async(download_images_for_city, (city, lat, lon))

    pool.close()
    pool.join()

if __name__ == '__main__':
    download_images()
```

Листинг А.3 – Скрипт для генерации набора данных

Б. Рисунки и Иллюстрации

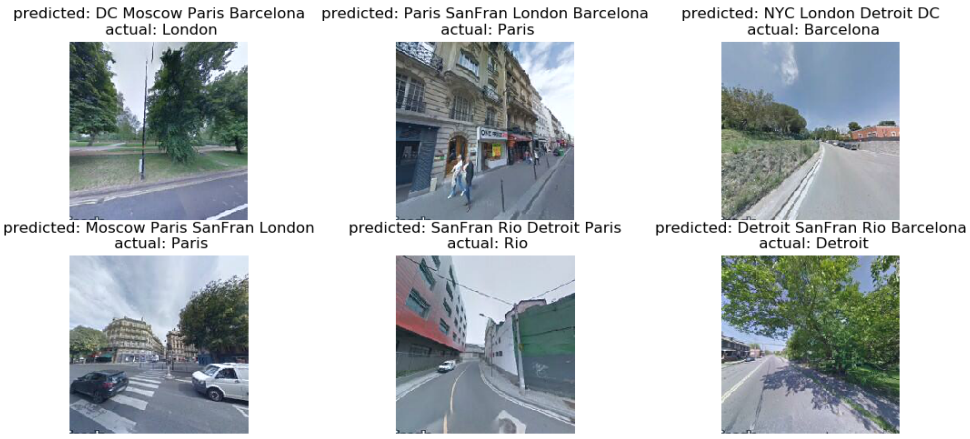


Рис. Б.1 – Дополнительная визуализация для случая с 1 фотографией

Класс	precision	recall	f1-score	support
Barcelona	0.35	0.32	0.34	100
DC	0.51	0.36	0.42	100
Detroit	0.57	0.70	0.63	100
London	0.46	0.46	0.46	100
Moscow	0.36	0.48	0.41	100
NYC	0.47	0.29	0.36	100
Paris	0.30	0.31	0.31	100
Rio	0.45	0.43	0.44	100
SanFran	0.46	0.49	0.48	100
Sydney	0.54	0.63	0.58	100
avg / total	0.45	0.45	0.44	1000

Таблица Б.1 – Отчёт по классификации для полностью переученной сети на валидационной выборке

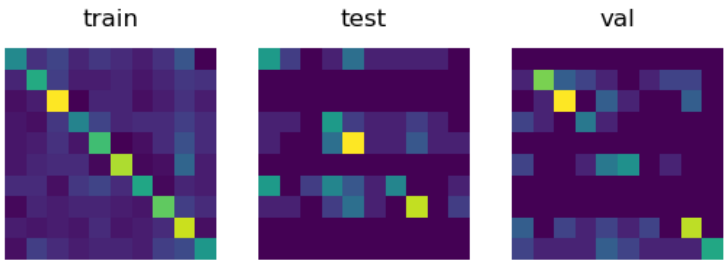


Рис. Б.2 – Матрицы неточностей для различных выборок.

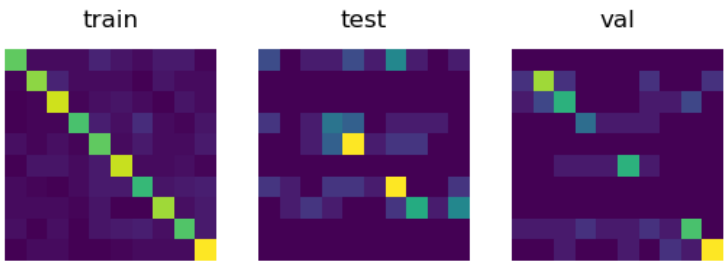


Рис. Б.3 – Матрицы неточностей для различных выборок для полностью переученной сети