



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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

на тему:

**«Классификация документов, удостоверяющих
личность, по фотографии»**

Студент ИУ7-83Б
(Группа)

А.Г. Сиденко
(И.О.Фамилия)

Руководитель ВКР

Д.С. Бабарыкин
(И.О.Фамилия)

Консультант

(И.О.Фамилия)

Нормоконтролер

(И.О.Фамилия)

2021 г.

РЕФЕРАТ

Отчёт содержит 51 страниц, 15 рисунка, 3 таблицы, 43 источников, 1 прил.

СОДЕРЖАНИЕ

Введение	8
1 Аналитический раздел	10
1.1 Постановка задачи	10
1.2 Существующие решения	11
1.3 Классификация документов по визуальным признакам	13
1.3.1 Выбор модели нейронной сети	13
1.4 Обзор и анализ методов распознавания и классификации текстовой информации	16
1.4.1 Оптическое распознавание символов	16
1.4.2 Индексация текста	20
1.4.3 Методы классификации	23
1.5 Вывод	27
2 Конструкторский раздел	29
2.1 Функциональная модель	29
2.2 Предобработка изображения и классификация по визуальным признакам	30
2.3 Размер выборки	33
2.4 Извлечение текста из изображения, его предобработка и классификация	33
2.5 Объединение	35
2.6 Выборка	36
2.7 Вывод	37
3 Технологический раздел	38
3.1 Входные и выходные данные	38
3.2 Язык программирования	38
3.2.1 Библиотеки	38

3.3 UML диаграмма разработанного ПО	39
3.4 Пользовательский интерфейс	41
3.5 Вывод	42
4 Исследовательский раздел	43
Заключение	44
Список использованных источников	45
Приложение А Листинги	49

ВВЕДЕНИЕ

Правильная организация работы с документами в наши дни имеет большое значение, так как от эффективности реализации документооборота напрямую зависит эффективность работы любой организации. Например, в рамках кредитных конвейеров юридических лиц банки требуют от компаний предоставлять оригиналы различных документов. Для удобства использования следует классифицировать как отдельные документы, включая одностораничные или многостраничные документы [1].

Во многих интернет-сервисах, от платежных систем до схем восстановления доступа к учетным записям в социальных сетях, активно используются инструменты распознавания документов.

Например, в таких системах, как портал «Госуслуги» и платежная система «Webmoney», целью обработки документа является получение всей персональной информации о клиенте, например, в случае паспорта РФ это фамилия, имя, серия и номер паспорта, место и дата выдачи, код подразделения, машиночитаемая зона, и установление ее подлинности.

В социальных сетях, таких как «ВКонтакте», для восстановления доступа к учетной записи пользователя, не обязательно получать всю персональную информацию, указанную на странице документа, удостоверяющего личность пользователя. В большинстве случаев достаточно проверить изображение определенной части документа, содержащей фотографию пользователя. Данная часть должна полностью содержать фотографию пользователя, имя, фамилию.

Типы документов определяются текстовой и визуальной информацией. Например, паспорт или трудовую книжку легко визуально отличить, не анализируя текст в ней. Кроме того, если вы используете неспециализированное решение, качество распознавания текста в таких документах довольно низкое. Поэтому визуальный компонент несет в себе более релевантную информацию для классификации. Однако различные типы шенгенских виз могут быть визуально похожи, но они содержат различную текстовую информацию.

Таким образом, задача классификации документов сводится к модели, которая должна сочетать в себе два неструктурированных источника данных: визуальное представление документа и результат распознавания текстовой информации.

Целью работы является разработка и исследование метода классификации документов по фотографии. Имеется множество документов написанных на естественном языке и множество заранее известных категорий. Требуется для каждого документа выбрать категорию, к которой он, в силу своего смыслового (семантического) содержания, относится с наибольшей долей уверенности.

Для достижения поставленной цели необходимо решить следующие задачи.

- Анализ существующих решений.
- Разработка метода классификации.
- Спроектировать и реализовать ПО, демонстрирующее работу метода.
- Провести исследование на применимость данного метода, его соответствие цели работы.

1 Аналитический раздел

В данном разделе рассматривается предметная область, проводится постановка задачи, производится анализ подходов к реализации различных этапов.

1.1 Постановка задачи

Классификация документов является одной из задач информационного поиска, которая заключается в определении документа к одной из нескольких категорий на основе его содержания.

Задачу классификации типа удостоверения личности на изображении, рассматриваемую в данной работе, можно сформулировать следующим образом.

Существует множество документов, удостоверяющих личность, а также множество заранее известных классов документов. Изображение удостоверения личности Q должно быть отнесено к одному из классов $C = C_i, i \in [0, N]$ с определенной вероятностью.

Данное программное обеспечение предоставляет возможность классификации документов следующих типов.

1. Паспорт гражданина Российской Федерации.
 - 1) Первая страница – когда и кем выдан.
 - 2) Вторая страница – фотография и личные данные
 - 3) Третья страница – прописка.
2. Водительское удостоверение.
 - 1) Образец №1 – ламинированное бумажное с 1995 по 2011.
 - 2) Образец №2 – пластиковое с 1995 по 2011.
 - 3) Образец №3 – удостоверение нового образца.
3. Загранпаспорт гражданина Российской Федерации.

4. Шенгенская виза.

1) Франция.

2) Германия.

3) Италия.

4) Испания.

Классификатор получает входные данные: фотографию документа. На выходе классификатор сообщает, что он получил (паспорт, водительское удостоверение и так далее).

1.2 Существующие решения

Необходимо рассмотреть существующие решения и целесообразность создания нового. На рынке представлено множество разнообразных систем, занимающихся распознаванием документов, как платных, так и бесплатных.

1. **Smart ID Engine** [2].

2. **ABBYY** [3].

3. **IRIS** [4].

4. **Jumio** [5].

5. **Idmatch** [6].

Проведенное сравнение данных решений, представлено в таблице 1.1.

Таблица 1.1 — Сравнение существующих решений.

	Распознаваемые документы	Распознаваемые языки	Платформы	Доступность	Дополнительно
Smart ID Engine	Международные документы (паспорта, визы, водительские права)	Все основные	Android iOS Windows MacOSX	Платное	Сильно ориентируется на машинно-читаемую строку
ABBYY	Международные документы (паспорта, визы, водительские права)	Все основные	Android iOS Windows Linux	Платное	
IRIS	Документы США, международные паспорта и ID-карты	Нет поддержки русского	Windows MacOSX	Платное	
Jumio	Документы Европы и США	Нет поддержки русского	Android iOS	Платное	
Idmatch	Id-карты Киргизской Республики	Киргизский и русский языки	Нет готового ПО	Бесплатное	

Вывод по таблице

IRIS, Jumio, Idmatch – не идентифицируют необходимые документы. ABBYY – решение подходит только для мобильных устройств. Smart ID Engine

- плохо идентифицирует документы при закрытии машинно-читаемой строки.
Необходимо создать собственное решение.

1.3 Классификация документов по визуальным признакам

Далее необходимо рассмотреть этапы обработки изображения, а именно методы классификации по визуальным и текстовым признакам.

Для начала, рассмотрим способы классификации документов по визуальным признакам.

Оптимальным методом классификации являются **нейронные сети**.

Существует множество архитектур нейронных сетей, необходимо проанализировать и сравнить их.

1.3.1 Выбор модели нейронной сети

Задача классификации изображений является популярной и хорошо изученной. Выбор алгоритма классификации будет производится на базе результатов международного соревнования ILSVRC [7] (ImageNet Large Scale Visual Recognition Challenge). Соревнование оценивает алгоритмы обнаружения объектов и классификации изображений. Его целью является выявление самых лучших с точки зрения точности и скорости алгоритмов обнаружения и классификации. Оценка алгоритмов производится с помощью выборки данных ImageNet [8]. Эта выборка представляет из себя большую визуальную базу данных, содержащую более 14 миллионов изображений, которые в свою очередь разбиваются на 21841 категорию.

Начиная с 2012 года данное соревнованию выигрывают сверточные нейронные сети (Convolutional Neural Networks), а именно AlexNet [9], ZFNet [10], VGGNet [11], GoogLeNet [12], ResNet [13], TrimpNet (Не было никакого нового научного вклада, который оправдывал бы подготовку статьи, и по этой причине авторы TrimpNet только поделились результатами) и SENet [14]. С 2015 года CNN превзошли человеческие показатели – 5.1% [15].

1. AlexNet.

AlexNet была первой свёрточной нейронной сетью, выигравшей соревнование по классификации ImageNet в 2012 году. Архитектура AlexNet состоит из пяти свёрточных слоев, между которыми располагаются пулинг слои и слои нормализации, три полносвязанных слоя завершают нейронную сеть.

На схеме архитектуры все выходные изображения разделены на две одинаковые части – это связано с тем, что нейронная сеть обучается на старом GTX580, который имел всего 3 ГБ видеопамяти. Для обработки использовались две видеокарты, чтобы параллельно выполнять операции над двумя частями изображения.

2. **ZFnet.**

В 2013 году нейросеть ZFnet смогла достичь результата 11.7% – архитектура AlexNet использовалась в качестве основы, но с изменёнными параметрами и слоями.

3. **VGGnet.**

Появилась в 2014 году. Основная идея – использовать вместо больших сверток (11x11 и 5x5) маленькие свертки (3x3). С маленькими фильтрами мы получим не так много параметров, но при этом сможем гораздо эффективнее обрабатывать их.

4. **GoogleNet.**

GoogleNet или Inception-v1 – ещё более глубокая архитектура с 22 слоями. Целью Google было разработать нейросеть с наибольшей вычислительной эффективностью. Для этого они придумали так называемый модуль Inception – вся архитектура состоит из множества модулей, следующих друг за другом.

5. **ResNet.**

ResNet – это сокращенное название для Residual Network (дословно – «остаточная сеть»). Решает проблему деградации (простая сеть увеличивает частоту ошибок по мере того, как сеть углубляется), и после использования остатка, когда сеть углубляется, частота ошибок все еще может уменьшаться.

6. **SENet.**

Squeeze-and-Excitation Networks (SENets) представляют собой специальный блок для свёрточной нейронной сети, который улучшает взаимозависимости каналов практически без дополнительных вычислений. Основная идея – добавить параметры к каждому каналу свёрточного блока, чтобы сеть могла адаптивно регулировать вес каждой карты признаков.

В ImageNet основными метриками ошибок являются: top-1 и top-5, где частота ошибок top-5 – это доля тестовых изображений, для которых правильная метка не входит в число пяти меток, которые модель считает наиболее вероятными.

Результаты сравнения моделей представлены в таблице 1.2.

Таблица 1.2 – Сравнение моделей сверточных нейронных сетей.

Название сети [16]	Год	Ошибка Топ-5	Количество обучаемых параметров [17]	Требуемое оборудование [18]
AlexNet	2012	16.4 %	60 миллионов	2 GPU
ZFNet	2013	11.7 %	60 миллионов	1 GPU
VGGNet	2014	7.3 %	138 миллионов	4 GPU
GoogleNet	2014	6.7 %	4 миллиона	CPU
ResNet	2015	3.5 %	25.6 или 1.7 миллиона	2 GPU
TrimpNet	2016	2.99 %		
SENet	2017	2.25 %	27.5 миллионов	8 GPU

Вывод по таблице

В соответствии с результатами ILSVRC последних лет, можно отметить, что наилучшей точностью классификации объектов на сегодняшний день обладает свёрточная нейронная сеть SENet. Однако этот алгоритм является очень требовательным к вычислительным ресурсам.

Принято компромиссное решение – использовать модель нейронной сети GoogleNet, так как она дает хорошие показатели точности и не требует мощные вычислительные ресурсы.

1.4 Обзор и анализ методов распознавания и классификации текстовой информации

Второй составляющей метода является распознавание и классификация текстовой информации в документе.

Таким образом, процесс делится на три связанных между собой этапа:

- этап выделения текстовой информации из изображения;
- этап преобразования текста в вектор признаков, в процессе которого определяется наиболее полное и информативное представление текста в виде числового вектора;
- этап классификации, в процессе которого проверяется гипотеза принадлежности изображения классу изображений объекта на основании наблюдения (вектора признаков).

1.4.1 Оптическое распознавание символов

Преобразование графического изображения в текст выполняется специальными программами распознавания текста (Optical Character Recognition - OCR).

Основной принцип автоматического распознавания образов заключается в обучении машины распознавать все возможные эталонные образцы и сравнивать их с идентифицированными объектами. В системах распознавания это буквы, цифры и знаки препинания. Обучение осуществляется путем показа машине образцов символов различных классов. На основе этих образцов машина генерирует прототип описания каждого класса объекта. Затем в процессе распознавания неизвестные символы сравниваются с заранее полученными образцами (выборкой) и определяется класс, с которым обнаружено больше всего совпадений.

Системы оптического распознавания текста – OCR-системы технологии, которая позволяет преобразовывать различные типы документов, такие как отсканированные документы, PDF-файлы или фото с цифровой камеры, в редактируемые форматы с возможностью поиска [19].

OCR – это одно из направлений компьютерного зрения.

Современные системы оптического распознавания можно разделить на коммерческие и свободно распространяемые системы с открытыми исходными кодами.

Для сравнения интерес представляют обе системы, как коммерческие, ориентированные на высокое качество распознавания, так и открытые системы, ориентированные на доступность и гибкость конфигурации. Поскольку целью данной работы является работа с документами на языках: русском, английском, испанском, французском, итальянском, немецком, интерес представляют системы, поддерживающей распознавание данных языков.

ABBYY FineReader [20] – это программное обеспечение оптического распознавания символов (OCR). Поддерживает распознавание текста на 190 языках. OCR или распознавание текста, использует интеллектуальные алгоритмы для преобразования изображений в редактируемый текст с сохранением исходного макета и формата исходного документа. Является признанным лидером на рынке. Распространяется на коммерческой основе.

IRIS Readiris [21] – это программа для преобразования документов в различные форматы. Распознает более 130 языков, включая русский, принимает файлы и сохраняет результаты во всех возможных форматах. Программное обеспечение платное.

Cuneiform [22] – свободно распространяемая открытая система оптического распознавания текстов российской компании Cognitive Technologies. CuneiForm позиционируется как система преобразования электронных копий бумажных документов и графических файлов в редактируемый вид, способная сохранять структуру и тип шрифта исходного документа в автоматическом или полуавтоматическом режиме. Система включает в себя две программы для одиночной и пакетной обработки электронных документов. Поддерживает все необходимые нам языки: английский, русский, испанский, итальянский, французский, немецкий. Кроме того, поддерживается сочетание русского и английского языка.

Tesseract [23] – свободная компьютерная программа для распознавания текстов, разрабатывавшаяся Hewlett-Packard, а затем Google купил её и открыл исходный код под лицензией Apache 2.0 для продолжения разработки. Сегодня Tesseract считается одним из самых мощных решений с открытым исходным кодом для распознавания данных отсканированных документов. Tesseract поддерживает более 100 различных языков, что делает его универсальным и широко используемым решением во всём мире. Многие технологические компании создают комплексные интеллектуальные решения для обработки данных, в основе которых лежит Тессеракт.

OCRFeeder [24] – программа, предоставляющая графический интерфейс пользователя для систем оптического распознавания символов CuneiForm, Tesseract, GOCR и Ocrad. OCRFeeder является свободно распространяемой программой для операционной системы Linux.

Выбор OCR систем основывался на сравнительном анализе их результатов по трем типам данных (среднее качество, высокое качество и очень высокое качество) [25]. Примеры изображений приведены на рисунках 1.1, 1.2, 1.3.

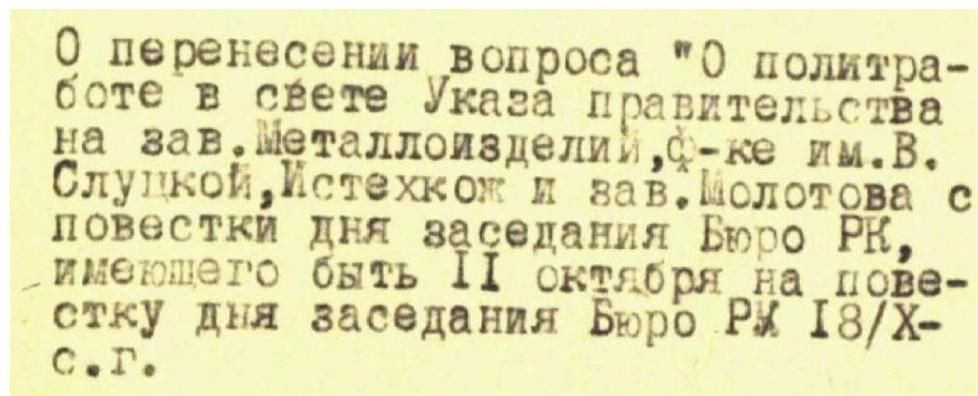


Рисунок 1.1 – Среднее качество.

О выполнении постановления бюро РК КПСС от 9 марта 1973 года "О работе партийных организаций сельскохозяйственных предприятий по повышению роли специалистов в развитии общественного производства в свете постановления ЦК КПСС по Алтайской краевой партийной организации".

Рисунок 1.2 — Высокое качество.

1999 год
Документы (договоры, соглашения, справки, переписка) о передаче средств федерального бюджета для закупки и поставки сельскохозяйственной продукции в Федеральный продовольственный фонд, в Федеральный фонд зерна, о финансировании ярмарки «Российский фермер»

Рисунок 1.3 — Очень высокое качество.

Качество результатов распознавания представлено в таблице 1.3.

Таблица 1.3 — Сравнение качества распознавания текста.

	Качество распознавания документов среднего качества	Качество распознавания документов высокого качества	Качество распознавания документов очень высокого качества	Доступность
ABBYY FineReader	76,8%	90,55%	99,25%	коммерческое
IRIS Readiris	22,01%	68,34%	94,40%	коммерческое
Cuneiform	0%	30,28%	90,20%	open-source
Tesseract	36,06%	74,68%	94,13%	open-source

Вывод по таблице

Результаты сравнительного анализа показывают, что коммерческая система «Abby Finereader» достигает максимального уровня качества. В свободно распространяемых системах лучшим является – Tesseract.

1.4.2 Индексация текста

Индексация документов – это получение вектора признаков для текста, состоит из двух этапов: получение термов и взвешивание термов.

Модели перевода текста в векторное пространство [26].

1. Мешок слов. В данной модели порядок слов не имеет значения, все документы представлены матрицей, где строка – это документ, столбец – слово. Элемент на пересечении строки и столбца – вес слова в документе.

2. Мешок N-грамм. Часто информацию несут не только отдельные слова, но некоторые последовательности. В данной модели учитываются вместо слов последовательности из N слов. Все документы будут также представлены матрицей.

3. Word2vec. Для каждого слова находится его векторное представление, которое содержит информацию о контекстных словах. После чего применяется метод кластеризации и в новом виде тексты состоят не из слов, а из номеров кластеров.

Далее необходимо рассмотреть различные функции взвешивания [27].

1. Частотные векторы. Заполняем вектор частотой, с которой слово появляется в документе.

Например, на рисунке 1.4 представлено, как бы выглядел вектор для текста номер 2.



Рисунок 1.4 — Функция взвешивания, частотные векторы

2. Унитарное кодирование. Заполняем вектор значениям 1 или 0, в зависимости от того есть ли слово в документе, или его нет, соответственно.

Например, на рисунке 1.5 представлено, как бы выглядел вектор для текста номер 1.



Рисунок 1.5 — Функция взвешивания, унитарное кодирование

3. Частота слова – обратная частота документа (TF_IDF) [28].

TF_IDF нормализует частоту токенов в документе по отношению к остальной части. Подход подчеркивает термины, которые часто встречаются в пределах данного документа и резко употребляются в других.

Частота термина TF – оценка важности слова t в пределах одного документа d , вычисляется по формуле 1.1.

$$TF = \frac{n_{t,d}}{n_d} \quad (1.1)$$

Обратная частота документа IDF – инверсия частоты, с которой слово t встречается в документах коллекции, вычисляется по формуле 1.2.

$$IDF = \log \frac{|D|}{D_t} \quad (1.2)$$

Итоговый вес слова в документе относительно всех документов вычисляется по формуле 1.3

$$V_{t,d} = TF \cdot IDF \quad (1.3)$$

Например, на рисунке 1.6 представлено, как бы выглядел вектор для текста номер 3.

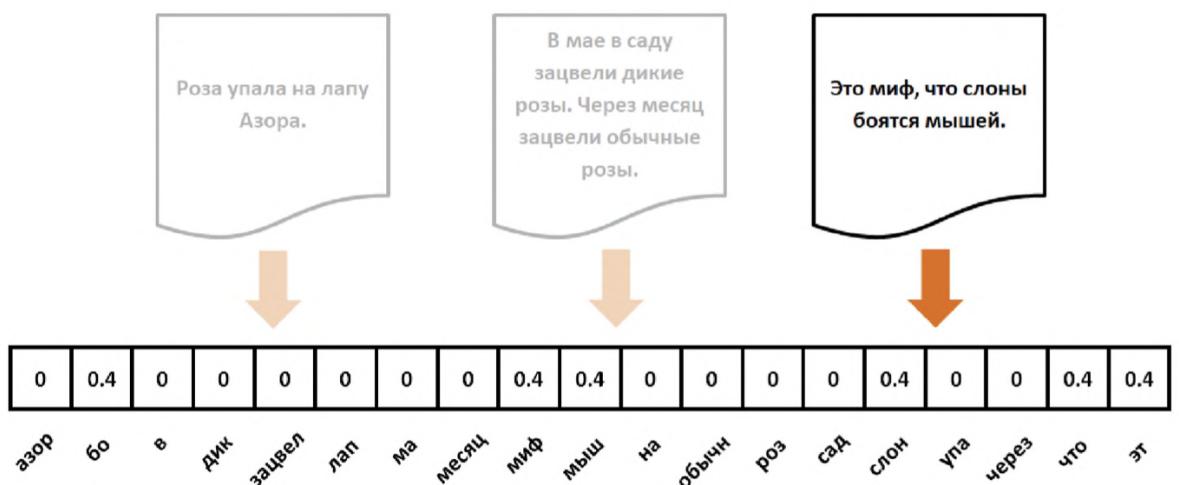


Рисунок 1.6 — Функция взвешивания, TF_IDF

Вывод по сравнению методов индексации текста

В данной работе последовательность слов и их семантика значения не имеет, поэтому было приятно решение использовать модель – мешок слов. Весовой функцией выбрана – TF_IDF, так как в документах встречаются

слова, которые появляются только в одном, а значит более важны. А есть слова, которые встречаются во многих и их значимость будет меньше.

1.4.3 Методы классификации

Существует множество методов классификации, которые используют различный математический аппарат и различные подходы при реализации. Однако эффективность этих методов зависит от конкретной решаемой задачи. Несмотря на то, что в течение последнего десятилетия коммерческие компании занимаются проблемой повышения качества машинного обучения, на сегодняшний день не существует методов, которые могли бы однозначно эффективно решить задачу классификации.

Можно выделить следующие типы методов классификации: вероятностные, метрические, логические, линейные, логическая регрессия [28]. Необходимо обобщенно описать некоторые из них, указывая преимущества и недостатки каждого из них.

1. Метод Байеса.

Метод Байеса (Naive Bayes, NB) относится к вероятностным методам классификации [29].

Пусть имеется множество классов $C = \{c_0, c_1, \dots, c_N\}$ документов. Согласно теореме Байеса вероятность того, что документ принадлежит классу c , имеет вид 1.4.

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \quad (1.4)$$

Вероятность $P(d)$ не требует вычислений ввиду того, что его значение не зависит от класса c , а значит, не влияет на нахождение наибольшей вероятности.

Преимущества метода состоят в следующем:

- высокая скорость работы;
- поддержка инкрементного обучения;

- простая реализация алгоритма в виде программы,;
- легкая интерпретируемость результатов работы алгоритма.

Несмотря на приведенные достоинства, метод Байеса имеет так же и минусы в своей реализации.

- Относительно низкое качество классификации.
- Неспособность учитывать зависимость результата классификации от сочетания признаков являются главными недостатками этого метода.

2. Метод k ближайших соседей.

Метод k ближайших соседей (k Nearest Neighbors, KNN) относится к метрическим методам и считается простейшим классификатором [30]. Суть метода заключается в том, что документу d присваивается тот класс c , к которому принадлежит большинство из k ближайших соседей документа, вычисленных с помощью какой-либо метрики расстояния.

Достоинства данного метода:

- простая реализация;
- легкая интерпретируемость.
- возможность обновления обучающей выборки без переобучения классификатора;
- устойчивость алгоритма к аномальным выбросам в исходных данных;
- хорошее обучение в случае с линейно неразделимыми выборками.

К недостаткам относятся:

- недостаточная производительность в реальных задачах, так как число соседей, используемых для классификации, будет достаточно большим;
- трудность в наборе подходящих весов и определением, какие признаки необходимы для классификации;

- высокая зависимость от выбранной метрики расстояния между примерами;
- невозможность решения задач большой раз-мерности по количеству классов и документов.

3. Метод деревьев решений.

Метод деревьев решений (Decision Trees, DT) относится к логическим методам классификации [31]. Дерево решений представляет собой конечный связный граф с множеством вершин, по которому производится классификация документов, описанных набором признаков. В узлах (вершинах) дерева прописаны условия, после проверки которых выполнение алгоритма классификации продолжается по правому или левому поддереву рассматриваемого узла. Процедура повторяется в каждом посещенном узле до тех пор, пока очередной узел не окажется листом. У каждого узла столько ветвлений, сколько значений имеет выбранный признак.

Метод обладает рядом преимуществ, таких как:

- простая программная реализация;
- быстрота обучения;
- высокая точность прогнозирования.

Однако также присущи и недостатки, такие как:

- неустойчивость алгоритмов к выбросам;
- необходим большой объем тренировочных данных для получения точных результатов.

4. Метод опорных векторов.

Метод опорных векторов (Support Vector Machine, SVM) является линейным методом классификации, в настоящее время призван одним из лучших [32]. Главным принципом SVM является определение раз-делителя в искомом пространстве, который разделяет классы наилучшим образом.

Преимущества метода:

- один из наиболее качественных методов;
- возможность работы с небольшим набором данных для обучения;
- сводимость к задаче выпуклой оптимизации, имеющей единственное решение.

Недостатки метода:

- сложная интерпретируемость параметров алгоритма;
- неустойчивость по отношению к выбросам в исходных данных.

5. Логистическая регрессия.

Одним из ранних применений регрессии к классификации текстов является линейный метод наименьших квадратов (ЛМНК) [33].

Преимущества метода:

- является одним из наиболее качественных;
- поддерживает инкрементное обучение;
- имеет относительно простую программную реализацию алгоритма.

Недостатки метода:

- сложная интерпретируемость параметров алгоритма;
- неустойчивость по отношению к выбросам в исходных данных.

6. Нейронные сети.

Нейронные сети состоят из набора «нейронов», которые являются преобразователями входных сигналов в выходные [34]. Преобразование задается весами сети, которые являются параметрами и могут изменяться. Выходные сигналы вычисляются как функция от входных сигналов. В общем случае нейронная сеть строится как соединение множества нейронов, объединенных в уровни, при этом выходы одного уровня являются входами следующего. Самая простая нейронная сеть состоит из одного слоя, однако круг решаемых такими сетями задач ограничен. Поэтому на практике часто используют нейронные сети, содержащие большее число слоев.

Преимущества метода:

- имеет очень высокое качество алгоритма при удачном подборе параметров;
- является универсальным аппроксиматором непрерывных функций;
- поддерживает инкрементное обучение.

Недостатки метода:

- вероятность возможной расходимости или медленной сходимости, поскольку для настройки сети используются градиентные методы;
- необходимость очень большого объема данных для обучения, чтобы достичь высокой точности;
- низкая скорость обучения;
- сложная интерпретируемость параметров алгоритма.

Вывод по анализу методов классификации текстов

Так как в работе важна точность, при этом без больших нагрузок системы. Для классификации текстовой информации выбран метод опорных векторов.

1.5 Вывод

В данном разделе выполнена постановка задачи работы, проанализированы существующие решения и необходимость создания нового. Были рассмотрены методы классификации документа по визуальным и текстовым признакам.

Исходя из приведенного сравнительного анализа были выбраны следующие подходы:

- В качестве модели нейронной сети, для классификации документов по визуальным признакам, была выбрана – googlenet.
- Для оптического распознавания текста – tesseract.

- Индексация текста – мешок слов и весовая функция – частота слова – обратная частота документа (TF_IDF).
- Для классификации текстовой информации – метод опорных векторов.

2 Конструкторский раздел

Цель данной работы – определить, какой тип документа загрузил пользователь.

Классификатор получает входные данные: фотографию документа. На выходе классификатор сообщает, что он получил (паспорт, водительское удостоверение или какие-либо другие типы документов).

2.1 Функциональная модель

На рисунке 2.1 изображен нулевой уровень декомпозиции задачи.

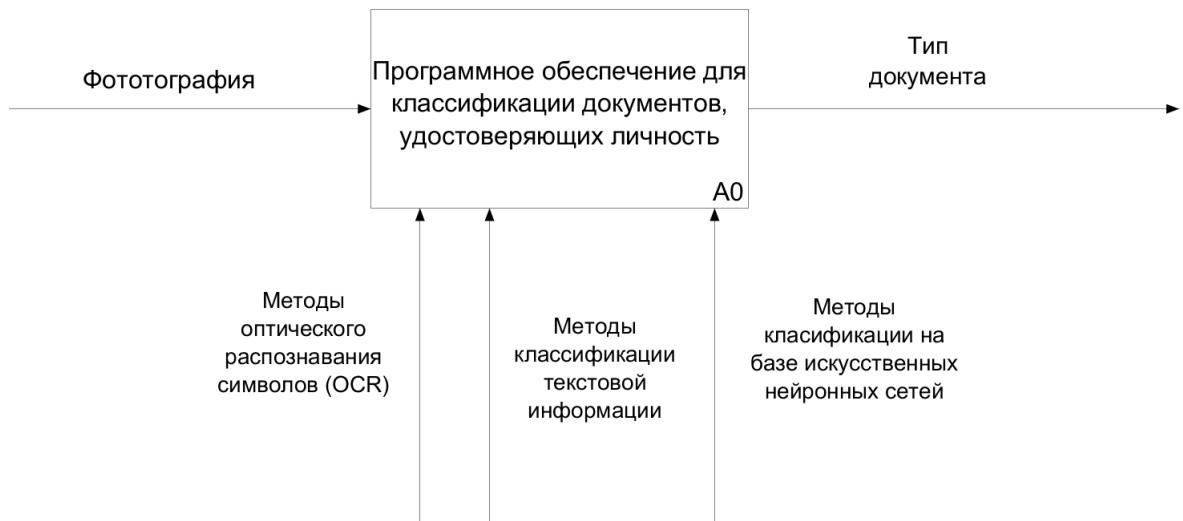


Рисунок 2.1 — Нулевой уровень декомпозиции задачи.

На вход классификатору поступает изображение, которое после предобработки поступает на вход классификатора по визуальным признакам. Также изображение поступает на вход системы оптического распознавания символов и после предобработки, текст поступает на вход классификатора по текстовым признакам.

В данной работе будет представлен алгоритм объединения двух классификаторов.

На основе выделенных этапов, был предложен подход к декомпозиции разрабатываемого метода, приведенный на рисунке 2.2.

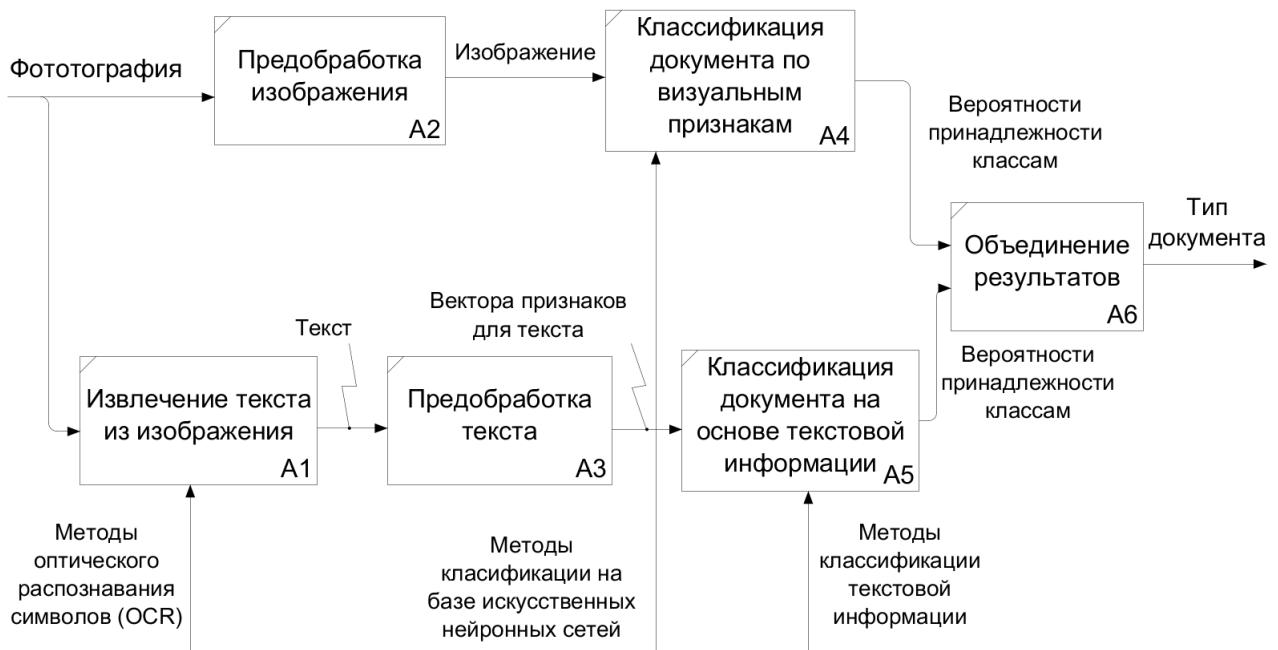


Рисунок 2.2 – Функциональная модель первого уровня.

Далее необходимо рассмотреть каждый из этапов.

2.2 Предобработка изображения и классификация по визуальным признакам

Свёрточная нейронная сеть googlenet принимает изображения формата 224x224, поэтому изначально изображения нужно привести к исходному формату.

На рисунке 2.3 представлена модель данной нейронной сети.

Она состоит из следующих слоев.

- Сверточный слой – применение операции свертки к выходам с предыдущего слоя, где веса ядра свертки являются обучаемыми параметрами.
- Пулинговый слой призван снижать размерность изображения. Исходное изображение делится на блоки размером $w \times h$ и для каждого блока вычисляется некоторая функция. В данной модели, в основном, используются функции максимума (max pooling) и среднего (average pooling).
- Inception – это специальный слой нейронной сети, впервые представленный в googlenet. Данный слой является небольшой локальной сетью и его

задача состоит в параллельном применении нескольких фильтров на исходное изображение, которые затем объединяются в один выходной. Данный слой позволяет сохранить малое число слоев, с сохранением полезной информации о изображении.

- Двумя проблемами в обучении глубоких нейронных сетей являются исчезающий градиент и взрывающийся градиент. Для борьбы с этой проблемой был предложен слой исключения (residual block). Идея заключается в том, чтобы взять пару слоёв (например, свёрточных), и добавить дополнительную связь, которая проходит мимо этих слоёв.
- Softmax – функция, которая превращает вектор действительных чисел в вектор вероятностей.

Все скрытые слои используют функцию активации ReLU.

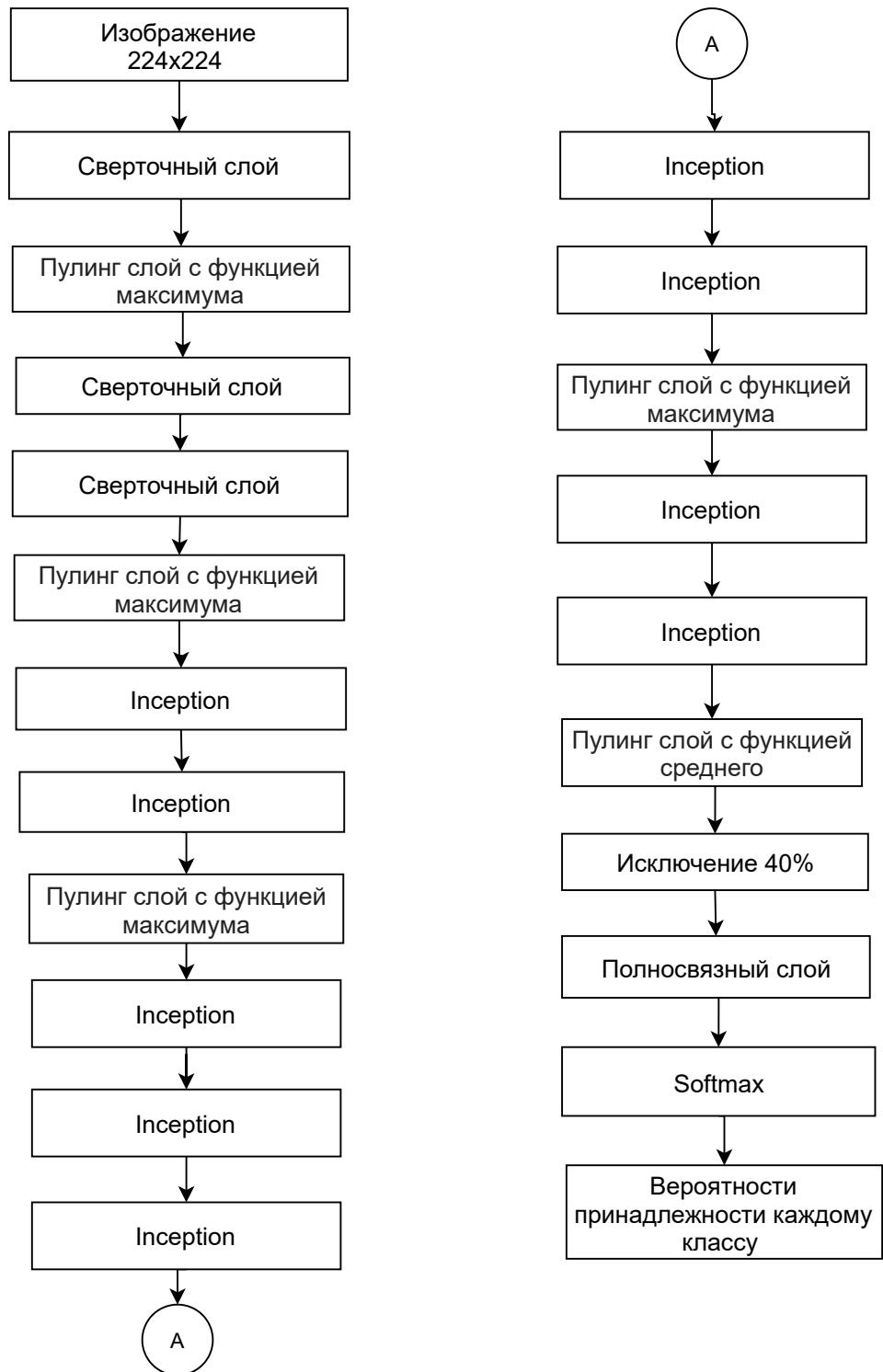


Рисунок 2.3 – Схема модели нейронной сети googlenet.

На выходе получаем вероятности принадлежности документа к каждому из классов, необходимо выбрать максимальную.

2.3 Размер выборки

Перед началом обучения необходимо определить размер выборки. Размер выборки обусловлен точностью определения параметров метода и необходимой сложностью исследования. Для оценки размера выборки можно воспользоваться законом больших чисел, записанном в форме 2-ого неравенства Чебышева, формула 2.1. [35]

$$P = \{ |X - MX| \geq \varepsilon \} \leq \frac{\sigma^2}{N\varepsilon^2} \quad (2.1)$$

N – размер выборки.

X – случайная величина.

ε – требуемая точность.

σ – дисперсия случайной величины.

При достоверности метода $P = 0,9$, а точность $\varepsilon = 0.05$, тогда $N = 1000$.

Данная нейронная сеть обучалась на выборке в 800 фотографий, из них страницы паспорт РФ и загранпаспорт – примерно по 120 изображений на каждый класс, водительские удостоверения – по 60 на каждый класс, визы представлены самым меньшим количеством данных – менее 30 на каждый класс.

2.4 Извлечение текста из изображения, его предобработка и классификация

Как говорилось ранее извлечение текста из изображения проходит с помощью методов оптического распознавания символов, а именно tesseract.

Для возможности дальнейшей классификации, текст необходимо предобработать.

- Во-первых, для уменьшения его объема (очистка от пунктуации и пустых строк).

- Во-вторых, для улучшения точности значимости оценки слов, необходимо очистить текст от стоп-слов – союзов, предлогов и прочего.
- Также необходимо привести все слова к одному регистру, чтобы можно было сравнивать их.
- После этого можем провести векторизацию текста для получения вектора слов и их весов.

Схема алгоритма предобработки текста представлена на рисунке 2.4.



Рисунок 2.4 — Схема алгоритма предобработки текста.

Таким образом, после выполнения данного этапа мы получаем векторное представление нашего текста.

Можно показать облако слов для классифицируемых в данной работе документов, представлено на рисунке 2.5.

Облако слов или тегов – это визуальное представление ключевых слов.



Рисунок 2.5 – Облако слов документов.

После получения векторного представления нашего текста, можно классифицировать его методом опорных векторов. Обучающая выборка такая же, как и для googlenet.

2.5 Объединение

Следующий этап – объединение результатов.

Ансамбль методов использует несколько обучающих алгоритмов с целью получения лучшей эффективности прогнозирования, чем могли бы получить от каждого обучающего алгоритма по отдельности.

Будет проходить с помощью взвешенного голосования.

Рассмотрим задачу, имеем 11 классов: $Y = \{1, 2, \dots, K\}$, $K = 11$.

И 2 классификатора или эксперта: f_1, \dots, f_M , $M = 2$.

Тогда результат голосования будет представлен формулой 2.2, где α_{ik} – вес голоса i-ого эксперта для k-ого класса, $I(f_i(x) = k)$ – вероятность принадлежности документа x классу k по решению эксперта i.

$$f(x) = \max_{k=1..K} \sum_{i=1}^M \alpha_{ik} I(f_i(x) = k), \alpha_{ik} > 0 \quad (2.2)$$

Задача исследовательского раздела состоит в проведении многомерной оптимизации Пауэлла для определения оптимальных весовых коэффициентов для каждого класса, формула 2.3.

$$f = f(\alpha_{00}, \alpha_{01}, \dots, \alpha_{MK}) \quad (2.3)$$

Найти такие $\alpha_{00}, \alpha_{01}, \dots, \alpha_{MK}$, при которых $f \rightarrow \min$.

2.6 Выборка

Выборка представляет собой набор фотографий документов, на каждом из которых документ занимает более 80% площади документа и располагается правильно (угол поворота относительно любой оси координат меньше 15 градусов). Также документы на фотографиях выборки хорошо освещены, находятся в фокусе и не смазаны.

Пример фотографии из выборки приведен на рисунке 2.6.



Рисунок 2.6 — Пример водительского удостоверения из выборки.

2.7 Вывод

В данном разделе было проведено проектирование метода классификации документов, удостоверяющих личность, по фотографии.

Были описаны входные и выходные параметры каждого этапа. Так же была описана выборка данных, которая представляет собой набор фотографий.

Для объединения двух классификаторов было решено использовать ансамбль классификаторов, использующих взвешенное голосование.

3 Технологический раздел

В данном разделе описываются средства реализации, их преимущества, схема разработанного программного обеспечения, приводятся листинги исходного кода и изображения интерфейса пользователя.

3.1 Входные и выходные данные

Входными данными является фотография документа, удостоверяющего личность. Документ на фотографии должен быть хорошо освещен, находиться в фокусе, не смазан и занимать не менее 70% площади фотографии, поворот документа вокруг любой оси должен быть менее 15 градусов.

Выходные данные – название класса документа, к которому он относится.

3.2 Язык программирования

В качестве языка реализации разрабатываемого метода был выбран Python версии 3.7 [36]. Использование данного языка в области машинного обучения и нейронных сетей крайне популярно, что благоприятно сказывается на доступности документации и библиотек в данной предметной области.

3.2.1 Библиотеки

В разработанном ПО были использованы следующие библиотеки.

1. NLTK (Natural Language Toolkit) [37] – пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python. Была использована для предобработки текста.
2. Scikit-learn [38] является библиотекой для работы с машинным обучением. Была использована при реализации метода опорных векторов и при обучении ее и также googlenet.

3. PyTesseract [39] – это пакет Python для OCR, соответственно был использован для извлечения текста с изображений.

4. OpenCV [40] представляет собой самую популярную библиотеку для работы с компьютерным зрением, обработки изображений и набором сопутствующих алгоритмов.

В данной работе были использованы базовые структуры (матрицы изображений, вектора), а также модули загрузки изображений с диска.

5. TensorFlow [41] – это библиотека с открытым исходным кодом, предназначенная для машинного обучения. С помощью данной библиотеки, а точнее ее надстройкой Keras [42], была написана модель googlenet.

6. PyQt [43] – кроссплатформенный фреймворк для разработки ПО на языках C++, Python, Ruby и других. Qt был использован для создания интерфейса разработанного программного обеспечения.

3.3 UML диаграмма разработанного ПО

ПО было поделено на следующие основные модули.

- Классификатор по визуальным признакам.
- Классификатор по текстовым признакам.
- Модуль предобработки.
- Интерфейс.

Полная UML диаграмма разработанного ПО представлена на рисунке 3.1.

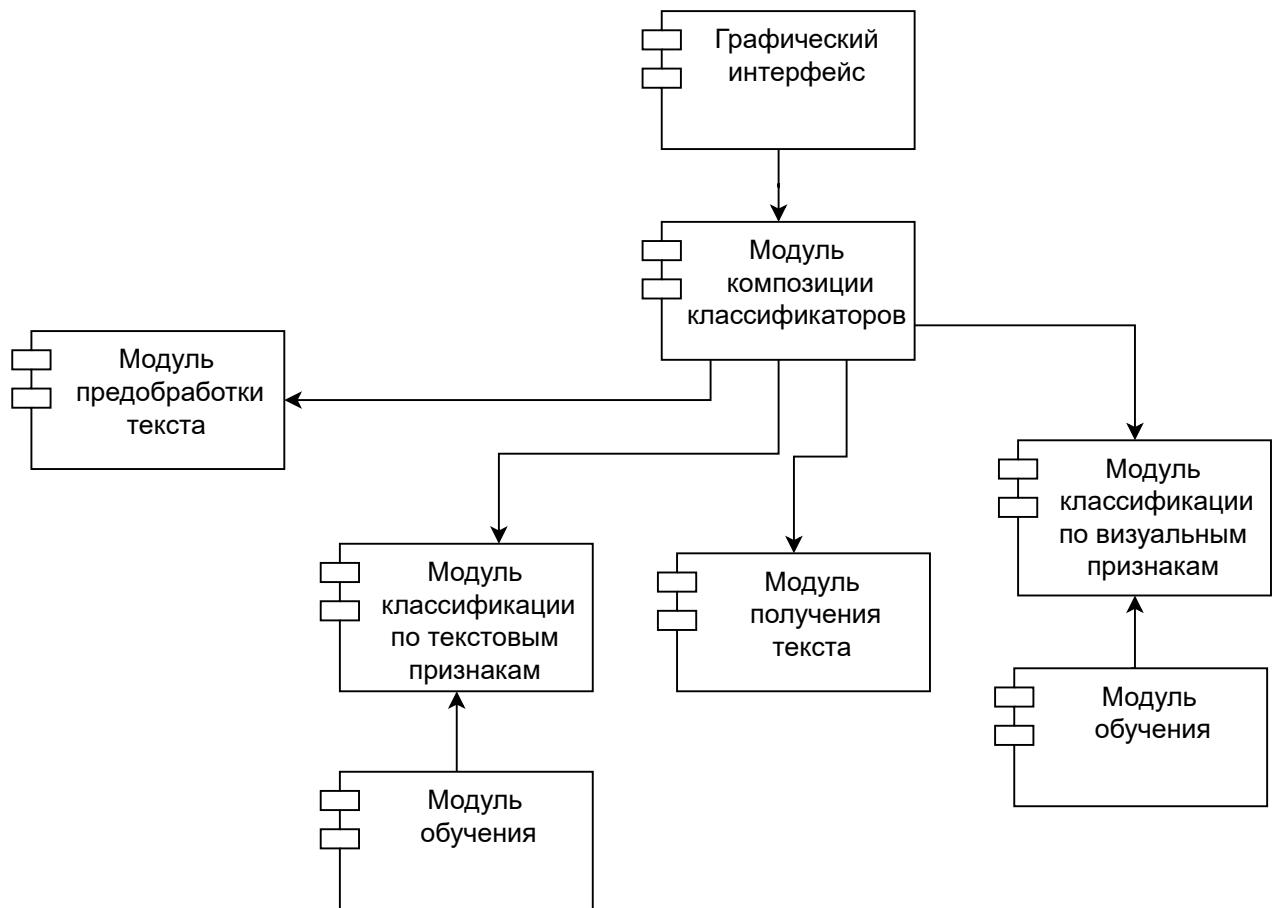


Рисунок 3.1 — UML диаграмма разработанного ПО.

Как было сказано выше для классификации документа по визуальным признакам была реализована модель googlenet, её программную реализацию можно посмотреть в листинге А.1.

После извлечения текста с изображения необходимо его предобработать перед классификацией, программная реализация алгоритма представлена в листинге А.2.

STOPWORDS – семантически незначимые слова, в данном случае необходимо рассмотреть слова из 6 языков, так как рассматриваются документы 6 стран.

Далее останется объединить классификаторы в ансамбль, код реализации представлен в листинге А.3.

3.4 Пользовательский интерфейс

Пример работы программы для двух типов документов – визы Германии и водительского удостоверения нового образца, представлен на рисунках 3.2, 3.3.

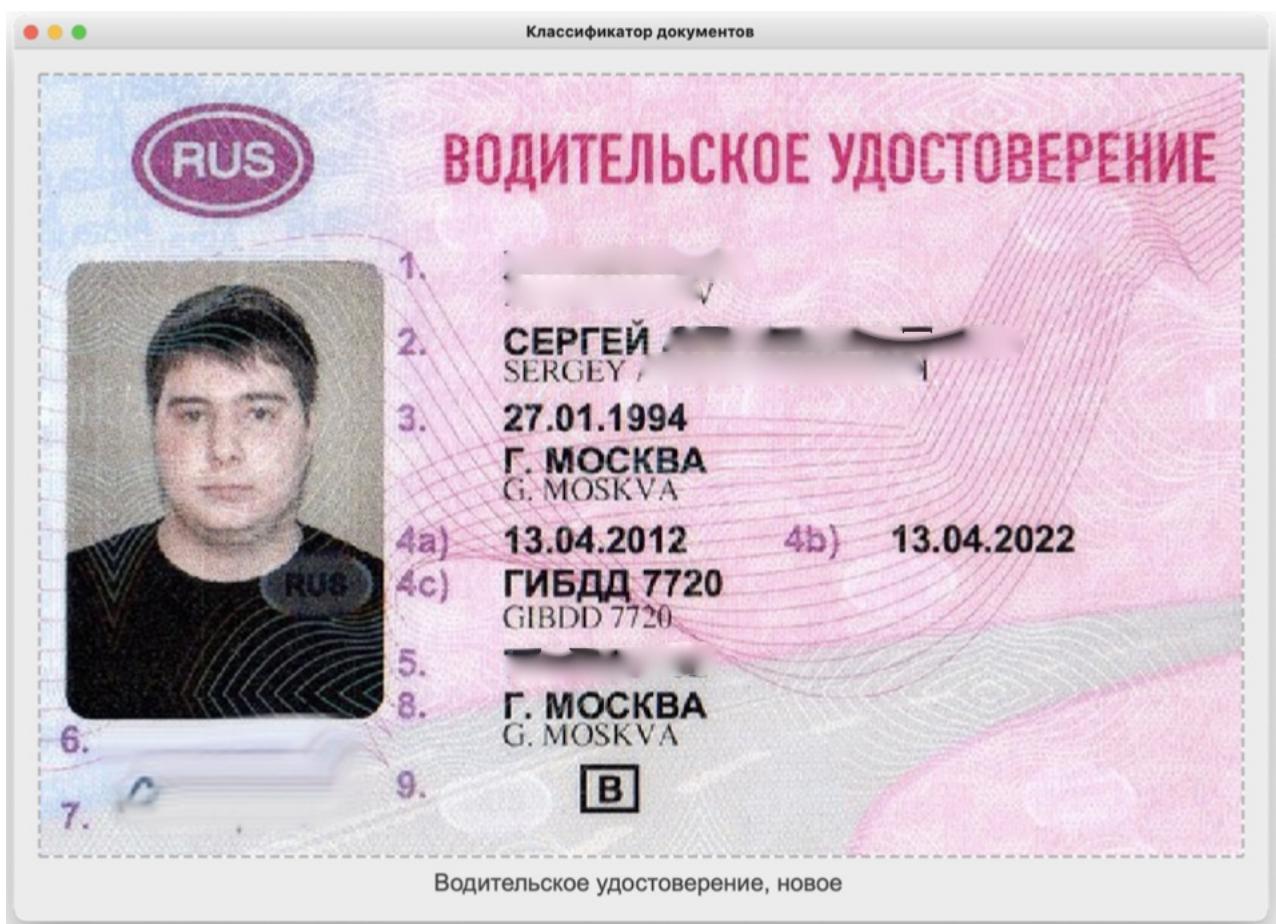


Рисунок 3.2 — Пример работы программы для типа документа – Водительское удостоверение нового образца.



Рисунок 3.3 – Пример работы программы для типа документа – Виза.

3.5 Вывод

В разделе проведены выбор средств реализации (язык программирования Python), а также используемых библиотеки (NLTK, Scikit-learn, PyTesseract, OpenCV, TensorFlow, PyQt), предоставлены данные о том, где использовались эти библиотеки. Был разработан метод, описан и разработан пользовательский интерфейс для просмотра результатов, модели классификаторов обучены. Также, были описаны входные (фотография с документом) и выходные (класс документа) данные.

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был разработан метод классификации документов по фотографии.

В ходе работы были выполнены все поставленные задачи.

- Произведен анализ существующих решений.
- Разработан метода классификации.
- Спроектировано и реализовано ПО, демонстрирующее работу метода.
- Проведено исследование на применимость данного метода, его соответствие цели работы.

Далее можно выделить следующие направления дальнейшего развития.

- Объединение разработанного метода классификации с методами сегментации, для получения подходящих изображений.
- Увеличение количества классов документов и точности классификации, путем увеличения выборки

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Использование алгоритмов ML для классификации многостраничных документов: опыт ВТБ. — Режим доступа: <https://habr.com/ru/company/vtb/blog/497484/> (дата обращения: 28.12.2020).
2. Smart ID Engine – Безопасное распознавание документов удостоверяющих личность 193 стран мира. — Режим доступа: <https://smartengines.ru/smart-idreader/> (дата обращения: 15.04.2021).
3. Abbyy – Извлечение данных из документов, удостоверяющих личность. — Режим доступа: <https://www.abbyy.com/ru/solutions/forms-processing/> (дата обращения: 15.04.2021).
4. IRIS – Manage and process easily official Identity documents. — Режим доступа: <https://www.irislink.com/PT/c1842/IRIS---Manage---process-easily-official-Identity-documents.aspx> (дата обращения: 15.04.2021).
5. Jumio Uses AI for Automatic Recognition of ID Documents. — Режим доступа: <https://www.jumio.com/ai-id-documents/> (дата обращения: 15.04.2021).
6. Idmatch – Распознавание документов. — Режим доступа: <https://idmatch.co> (дата обращения: 15.04.2021).
7. ImageNet Large Scale Visual Recognition Challenge. — Режим доступа: <https://www.image-net.org/challenges/LSVRC/indexo> (дата обращения: 15.04.2021).
8. Data for ILSVRC. — Режим доступа: <https://www.kaggle.com/c/imagenet-object-localization-challenge/data> (дата обращения: 15.04.2021).
9. Alex Krizhevsky Ilya Sutskever, Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks. — Springer, 2012.
10. Zeiler Matthew D, Fergus Rob. Visualizing and understanding convolutional networks. In European conference on computer vision. — Springer,

2014.

11. Simonyan Karen, Zisserman Andrew. Very deep convolutional networks for large-scale image recognition. — 2014.
12. Christian Szegedy Wei Liu Yangqing Jia Pierre Sermanet и др. Going deeper with convolutions. — 2015.
13. Kaiming He Xiangyu Zhang Shaoqing Ren, Sun Jian. Deep residual learning for image recognition. — 2016.
14. Jie Hu Li Shen, Sun Gang. Squeeze-and-excitation networks. — 2017.
15. What I learned from competing against a ConvNet on ImageNet. — Режим доступа: <http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/> (дата обращения: 17.04.2021).
16. Adversarial Machine Learning in Image Classification: A Survey Towards the Defender’s Perspective. — Режим доступа: <https://arxiv.org/pdf/2009.03728.pdf> (дата обращения: 15.04.2021).
17. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. — Режим доступа: <https://arxiv.org/pdf/1901.06032.pdf> (дата обращения: 15.04.2021).
18. Transfer Learning Using Convolutional Neural Network Architectures for Brain Tumor Classification from MRI Images. — Режим доступа: <https://europepmc.org/article/pmc7256397> (дата обращения: 15.04.2021).
19. Компьютерное зрение. — Режим доступа: <https://www.jetinfo.ru/computer-vision-technology-review/> (дата обращения: 28.12.2020).
20. ABBYY FineReader. — Режим доступа: <https://finereaderonline.com/ru-ru> (дата обращения: 17.04.2021).
21. Readiris 17. — Режим доступа: <https://www.irislink.com/EN-AU/c1729/Readiris-17--the-PDF-and-OCR-solution-for-Windows-.aspx> (дата обращения: 17.04.2021).

22. Cuneiform for Linux. — Режим доступа: <https://launchpad.net/cuneiform-linuxt> (дата обращения: 17.04.2021).
23. Tesseract Open Source OCR Engine. — Режим доступа: <https://github.com/tesseract-ocr/tesseract> (дата обращения: 17.04.2021).
24. OCRFeeder - A Complete OCR Suite. — Режим доступа: <https://gitlab.gnome.org/GNOME/ocrfeeder> (дата обращения: 17.04.2021).
25. Владимирович Смирнов Сергей. Технология и система автоматической корректировки результатов при распознавании архивных документов. — СПИИРАН, 2015.
26. Тлеубаев А.Т. Ступников С.А. Применение методов машинного обучения для автоматизации тематической разметки интернет-доменов. — Машинное обучение и анализ данных, 2018.
27. Федюшкин Н.А. Федосин С.А. О выборе методов векторизации текстовой информации. — Научно-технический вестник Поволжья №6, 2019.
28. Т.В. Батура. Методы автоматической классификации текстов. — Программные продукты и системы, 2017.
29. Наивный байесовский классификатор. — Режим доступа: <https://www.machinelearningmastery.ru/the-naive-bayes-classifier-e92ea9f47523/> (дата обращения: 28.12.2020).
30. Метод k ближайших соседей. — Режим доступа: <https://intuit.ru/studies/courses/6/6/lecture/176?page=3> (дата обращения: 28.12.2020).
31. Метод деревьев решений. — Режим доступа: <https://loginom.ru/blog/decision-tree-p1> (дата обращения: 28.12.2020).
32. Н.О. Баев. Использование метода опорных векторов в задачах классификации. — Международный журнал информационных технологий и энергоэффективности, 2017.

33. Линейные модели классификации и регрессии. — Режим доступа: <https://habr.com/ru/company/ods/blog/323890/> (дата обращения: 28.12.2020).
34. Применение нейронных сетей для задач классификации. — Режим доступа: <https://basegroup.ru/community/articles/classification> (дата обращения: 28.12.2020).
35. Печинкин А.В. Тескин О.И. Цветкова Г.М. и др. Теория вероятностей: Учеб. для вузов. — 3-е изд. — МГТУ им. Н.Э. Баумана, 2004.
36. Python. — Режим доступа: <https://www.python.org> (дата обращения: 10.05.2021).
37. Natural Language Toolkit. — Режим доступа: <https://www.nltk.org> (дата обращения: 10.05.2021).
38. Scikit-learn. — Режим доступа: <https://scikit-learn.org/stable/> (дата обращения: 10.05.2021).
39. PyTesseract. — Режим доступа: <https://pypi.org/project/pytesseract/> (дата обращения: 10.05.2021).
40. OpenCV. — Режим доступа: <https://opencv.org> (дата обращения: 10.05.2021).
41. Tensorflow. — Режим доступа: <https://www.tensorflow.org> (дата обращения: 10.05.2021).
42. Keras. — Режим доступа: <https://keras.io> (дата обращения: 10.05.2021).
43. PyQt. — Режим доступа: <https://doc.qt.io/qtforpython/> (дата обращения: 10.05.2021).

ПРИЛОЖЕНИЕ А

ЛИСТИНГИ

Листинг А.1 — Модель googlenet

```
1 def architecture(self):
2     input = Input(shape=(224, 224, 3))
3
4     layer = Convolution2D(filters=64,
5                            kernel_size=(7, 7),
6                            strides=2,
7                            padding='same',
8                            activation='relu')(input)
9     layer = MaxPooling2D(pool_size=(3, 3),
10                           strides=2,
11                           padding='same')(layer)
12
13    layer = Convolution2D(filters=64,
14                            kernel_size=(1, 1),
15                            strides=1,
16                            padding='same',
17                            activation='relu')(layer)
18    layer = Convolution2D(filters=192,
19                            kernel_size=(3, 3),
20                            strides=1,
21                            padding='same',
22                            activation='relu')(layer)
23    layer = MaxPooling2D(pool_size=(3, 3),
24                           strides=2,
25                           padding='same')(layer)
26
27    layer = self.Inception(input=layer,
28                            filters_1x1=64,
29                            filters_3x3_reduce=96,
30                            filters_3x3=128,
31                            filters_5x5_reduce=16,
32                            filters_5x5=32,
33                            filters_pool_proj=32)
34    layer = self.Inception(input=layer,
35                            filters_1x1=128,
36                            filters_3x3_reduce=128,
37                            filters_3x3=192,
38                            filters_5x5_reduce=32,
39                            filters_5x5=96,
40                            filters_pool_proj=64)
41    layer = MaxPooling2D(pool_size=(3, 3),
42                           strides=2,
43                           padding='same')(layer)
44
45    layer = self.Inception(input=layer,
46                            filters_1x1=192,
47                            filters_3x3_reduce=96,
48                            filters_3x3=208,
49                            filters_5x5_reduce=16,
50                            filters_5x5=48,
51                            filters_pool_proj=64)
```

```

53     aux1 = self.Auxiliary(layer)
54
55     layer = self.Inception(input=layer,
56                             filters_1x1=160,
57                             filters_3x3_reduce=112,
58                             filters_3x3=224,
59                             filters_5x5_reduce=24,
60                             filters_5x5=64,
61                             filters_pool_proj=64)
62     layer = self.Inception(input=layer,
63                             filters_1x1=128,
64                             filters_3x3_reduce=128,
65                             filters_3x3=256,
66                             filters_5x5_reduce=24,
67                             filters_5x5=64,
68                             filters_pool_proj=64)
69     layer = self.Inception(input=layer,
70                             filters_1x1=112,
71                             filters_3x3_reduce=144,
72                             filters_3x3=288,
73                             filters_5x5_reduce=32,
74                             filters_5x5=64,
75                             filters_pool_proj=64)
76
77     aux2 = self.Auxiliary(layer)
78
79     layer = self.Inception(input=layer,
80                             filters_1x1=256,
81                             filters_3x3_reduce=160,
82                             filters_3x3=320,
83                             filters_5x5_reduce=32,
84                             filters_5x5=128,
85                             filters_pool_proj=128)
86     layer = MaxPooling2D(pool_size=(3, 3),
87                           strides=2,
88                           padding='same')(layer)
89
90     layer = self.Inception(input=layer,
91                             filters_1x1=256,
92                             filters_3x3_reduce=160,
93                             filters_3x3=320,
94                             filters_5x5_reduce=32,
95                             filters_5x5=128,
96                             filters_pool_proj=128)
97     layer = self.Inception(input=layer,
98                             filters_1x1=384,
99                             filters_3x3_reduce=192,
100                            filters_3x3=384,
101                            filters_5x5_reduce=48,
102                            filters_5x5=128,
103                            filters_pool_proj=128)
104    layer = AveragePooling2D(pool_size=(7, 7),
105                           strides=1,
106                           padding='valid')(layer)
107
108    layer = Flatten()(layer)
109    layer = Dropout(rate=0.4)(layer)
110    layer = Dense(units=1000, activation='linear')(layer)
111    output = Dense(units=CLASS_NUM, activation='softmax')(layer)

```

```

112
113     return Model(inputs=input, outputs=[output, aux1, aux2])

```

Листинг А.2 — Предобработка текста

```

1 def text_preprocessing(text):
2     text = text.lower()
3     text_words_list = word_tokenize(text)
4
5     clear_words = []
6     for word in text_words_list:
7         if word not in STOPWORDS:
8             clear_words.append(word)
9
10    return str(clear_words)

```

Листинг А.3 — Ансамбль классификаторов

```

1 def predict_class(self, path):
2     img = mpimg.imread(path)
3     img = resize(img, (224, 224, 3))
4     img = img.reshape(1, 224, 224, 3)
5     out = self.model.predict(img)
6
7     predicted_label_visual = np.argmax(out[2])
8     predicted_proba_visual = out[2][0][predicted_label_visual] * \
9                               self.w1[predicted_label_visual]
10
11    text = get_all_text(path)
12    text_processed = text_preprocessing(text)
13    text_processed_vectorized = self.tfidf_vect \
14                                .transform([text_processed])
15
16    prediction_SVM = self.SVM.predict(text_processed_vectorized)
17    predicted_label_text = self.labelencoder \
18                           .inverse_transform(prediction_SVM)[0]
19    predicted_proba_text = self.SVM.predict_proba(
20                            text_processed_vectorized)[0] \
21                            [predicted_label_text] * \
22                            self.w2[predicted_label_text]
23
24    if (predicted_proba_text > predicted_proba_visual):
25        predicted_label = predicted_label_text
26    else:
27        predicted_label = predicted_label_visual
28
29    return self.documents.get(predicted_label)

```