МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University
Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

Хрипков Константин Михайлович

Разработка информационной системы распознавания образов на изображениях для решения задач классификации

ДИПЛОМНАЯ РАБОТА

Специальность 5В070300 – Информационные системы

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University
Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой КБОиХИ
канд. технических наук, доцент
_____Н. А. Сейлова
«____»_____20__г.

ДИПЛОМНАЯ РАБОТА

На тему: Разработка информационной системы распознавания образов на изображениях для решения задач классификации

Специальность 5В070300 – Информационные системы

Выполнил**:** Хрипков. К. М.

Научный руководитель

Магистр технических наук, тьютор

_____ Дуйсенбаева А. Н.

«____» _____ 20__ г

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University
Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

5В070300 – Информационные системы

УТВЕРЖДАЮ
Заведующий кафедрой КБОиХИ
канд. технических наук, доцент
_____Н. А. Сейлова
«____»_____20__г.

ЗАДАНИЕ
на выполнение дипломной работы

Обучающемуся: Хрипков Константин Михайлович
Тема: Разработка информационной системы распознавания образов на изображениях для решения задач классификации
Утверждена *приказом Ректора Университета № 762-б от 24.11.2020г.*
Срок сдачи законченной работы *27.05.2021г.*
Исходные данные к дипломному проекту: *результаты преддипломной практики, представление теоретического материала.*
Краткое содержание дипломной работы:
а) Анализ предметной области, постановка задач;
б) Инструменты и технологии;
в) Сбор датасета;
г) Проектирование системы;
д) Демонстрация работы системы;
е) Развитие и перспективы;
Рекомендуемая основная литература: *из 17 наименований*

ГРАФИК

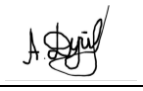
подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю**	**Примечание**
Анализ предметной области, постановка задач	06.04.2021г.	
Инструменты и технологии	06.04.2021г.	
Сбор датасета	16.04.2021г.	
Проектирование системы	30.04.2021г.	
Демонстрация работы системы	17.05.2021г.	
Развитие и перспективы	24.05.2021г.	

Подписи

консультантов и нормоконтролера на законченную дипломную работу (проект) с указанием относящихся к ним разделов работы (проекта)

Наименование разделов	**Консультанты, Ф.И.О. (уч. степень, звание)**	**Дата подписания**	**Подпись**
Разработка информационной системы распознавания образов на изображениях для решения задач классификации	Дуйсенбаева А. Н., магистр технических наук, тьютор		
Нормоконтролер	Кабдуллин М. А., тьютор		
Программная часть			

Научный руководитель:  Дуйсенбаева А. Н.

Задание принял к исполнению обучающийся:  Хрипков. К. М.

Дата "___" ноября 2020

StrikePlagiarism.com

SATBAYEV UNIVERSITY

Report date 6/1/2021
Edit date ---

Report has not been evaluated.

Metadata

Title

Разработка информационной системы распознавания образов на изображениях для решения задач классификации

Author(s)

Хрипков Константин Михайлович

Promoter

Асемгуль Дуйсенбаева

Organizational unit

ИКиИТ

List of possible text manipulation attempts

In this section, you can find information regarding text modifications that may aim at temper with the analysis results. Invisible to the person evaluating the content of the document on a printout or in a file, they influence the phrases compared during text analysis (by causing intended misspellings) to conceal borrowings as well as to falsify values in the Similarity Report. It should be assessed whether the modifications are intentional or not.

Characters from another alphabet	ß	1
Spreads	A→	0
Micro spaces	·	6
White characters	ß	0
Paraphrases (SmartMarks)	a	4

Record of similarities

Please note that high coefficient values do not automatically mean plagiarism. The report must be analyzed by an authorized person.

25
The phrase length for the SC 2

7587
Length in words

58569
Length in characters

Active lists of similarities

Scroll the list and analyze especially the fragments that exceed the SC 2 (marked in bold). Use the link "Mark fragment" and see if they are short phrases scattered in the document (coincidental similarities), numerous short phrases near each other (mosaic plagiarism) or extensive fragments without indicating the source (direct plagiarism).

The 10 longest fragments

Color of the text

NO	TITLE OR SOURCE URL (DATABASE)	NUMBER OF IDENTICAL WORDS (FRAGMENTS)	
1	https://oratorprofi.ru/raznoe/nvp-chto-eto-takoe-kak-rasschitat-npv-bystro-i-pravilno-formula-primery-instrukciya.html	48	0.63 %
2	https://www.ycombinator.com/library/4Q-a-minimum-viable-product-is-not-a-product-it-s-a-process	35	0.46 %
3	https://habr.com/ru/company/skillfactory/blog/525214/	27	0.36 %
4	https://oratorprofi.ru/raznoe/nvp-chto-eto-takoe-kak-rasschitat-npv-bystro-i-pravilno-formula-primery-instrukciya.html	20	0.26 %
5	**ДП Кунбулутова А, КочубинаА, Толочко В., науч рук Мерембаева А.С..docx** Кочубина А., Кунбулатова А., Толочко В. **4/6/2018** Almaty Management University (Deanery)	16	0.21 %

6	https://ru.wikipedia.org/wiki/Компьютерное_зрение	15	0.20 %
7	https://kpfu.ru/staff_files/F1493580427/NejronGafGal.pdf	10	0.13 %
8	https://obshaga.kz/page/mvup.to	10	0.13 %
9	https://www.klerk.ru/blogs/klerk/502818/	5	0.07 %

from RefBooks database (0.00 %)

NO	TITLE	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

from the home database (0.00 %)

NO	TITLE	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

from the Database Exchange Program (0.21 %)

NO	TITLE	NUMBER OF IDENTICAL WORDS (FRAGMENTS)	
1	**ДП Кунбулутова А, КочубинаА, Толочко В., науч рук Мерембаева А.С..docx** Кочубина А., Кунбулатова А., Толочко В. **4/6/2018** Almaty Management University (Deanery)	16 (1)	0.21 %

from the Internet (2.24 %)

NO	SOURCE URL	NUMBER OF IDENTICAL WORDS (FRAGMENTS)	
1	https://oratorprofi.ru/raznoe/nvp-chto-eto-takoe-kak-rasschitat-npv-bystro-i-pravilno-formula-primery-instrukciya.html	68 (2)	0.90 %
2	https://www.ycombinator.com/library/4Q-a-minimum-viable-product-is-not-a-product-it-s-a-process	35 (1)	0.46 %
3	https://habr.com/ru/company/skillfactory/blog/525214/	27 (1)	0.36 %
4	https://ru.wikipedia.org/wiki/Компьютерное_зрение	15 (1)	0.20 %
5	https://kpfu.ru/staff_files/F1493580427/NejronGafGal.pdf	10 (1)	0.13 %
6	https://obshaga.kz/page/mvup.to	10 (1)	0.13 %
7	https://www.klerk.ru/blogs/klerk/502818/	5 (1)	0.07 %

List of accepted fragments (no accepted fragments)

NO	CONTENTS	NUMBER OF IDENTICAL WORDS (FRAGMENTS)

АННОТАЦИЯ

Цель дипломной работы - разработка экспертной информационной системы, решающей задачу распознавания образов и классификации видов грибов и дающей пользователям рекомендации по сбору найденных образцов на основе обучения нейронных сетей.

При разработке системы было проведено исследование предметной области, по результатам которого были определены основные задачи и комплекс технологий для их решения. Были применены парадигмы data engineering, data science и глубокого обучения (deep learning). В ходе работы были задействованы инструменты веб-парсинга и преобразования данных, а также высокоуровневые API для машинного обучения в области компьютерного зрения.

Система использует собранные данные и метаданные видов грибов, распространенных в Казахстане и проводит классификацию с учетом разных регионов страны. Также она сопровождается планом развития, что в дальнейшем может быть использовано для реализации на её основе IT-продукта в гибкой парадигме MVP и выхода с этим продуктом на рынок.

АҢДАТПА

Дипломдық жұмыстың мақсаты - саңырауқұлақ түрлерін анықтау және классификациялау мәселелерін шешетін және пайдаланушыларға жүйке желілерін оқыту негізінде табылған үлгілерді жинауға ұсыныстар беретін сараптамалық ақпараттық жүйені құру.

Жүйені әзірлеу барысында пәндік саланы зерттеу жүргізілді, оның нәтижелері бойынша негізгі міндеттер мен оларды шешудің технологияларының жиынтығы анықталды. Data engineering, data science және deep learning парадигмалары қолданылды. Жұмыс барысында веб-талдау және деректерді түрлендіру құралдары, сонымен қатар компьютерлік көру саласында машиналық оқытуға арналған жоғары деңгейлі API қолданылды.

Жүйе Қазақстанда жиналған саңырауқұлақ түрлерінің жинақталған деректері мен метадеректерін қолданады және елдің әр түрлі аймақтарын ескере отырып жіктеу жүргізеді. Ол сонымен бірге даму жоспарымен бірге жүреді, оны кейінірек оның негізінде икемді MVP парадигмасында IT-өнімді енгізу және осы өніммен нарыққа шығу үшін пайдалануға болады.

THE ANNOTATION

The purpose of the thesis is to develop an expert information system that solves the problem of pattern recognition and classification of mushroom species and gives users recommendations for collecting found samples based on training neural networks.

During the development of the system, a study of the subject area was carried out, according to the results of which the main tasks and a set of technologies for their solution were determined. The paradigms of data engineering, data science and deep learning were applied. In the course of the work, web parsing and data transformation tools were used, as well as high-level APIs for machine learning in the field of computer vision.

The system uses the collected data and metadata of mushroom species common in Kazakhstan and carries out a classification taking into account different regions of the country. It is also accompanied by a development plan, which can later be used to implement an IT product on its basis in the flexible MVP paradigm and enter the market with this product.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Среди пяти царств живой природы отдельную позицию занимают грибы. Эти организмы включают в себя большую питательную ценность и полезны для здоровья. Конечно же, это так только в том случае, если эти грибы не токсичны и не ядовиты.

Сбор и употребление грибов сами по себе являются популярным занятием. Однако большинство людей, желающих собирать грибы, не имеет достаточно обширных знаний для их самостоятельной классификации. Классификация грибов необходима, чтобы избежать отравления или летального исхода. Таким образом, основная гипотеза заключается в том, что при сборе грибов целью человека является увеличение доли съедобных грибов в добыче, и соответственно уменьшение доли несъедобных или ядовитых.

Чтобы помочь пользователю достичь этой цели была разработана описанная в работе экспертная информационная система.

Экспертные системы позволяют заменить человека-специалиста в предметной области, в данном случае – профессионального грибника. Система принимает на вход изображение гриба, классифицирует вид гриба на нём, и даёт рекомендацию, стоит ли собирать его или нет. Классификация проводится моделью на основе архитектуры CNN. Модель способна распознавать 9 классов, после чего пользователю будет указан статус съедобности распознанного класса.

Точность классификации модели составляет ≈82%. Таким образом, 4 из 5 образов распознаются верно. Это подтверждает эффективность системы в целом, но также говорит и о том, что ответы системы носят рекомендательный характер, а не обязательный, и в случае выпуска системы как продукта, пользователи будут предупреждены, что все собранные по рекомендациям системы грибы подлежат дополнительной проверке.

Разработанная экспертная система относится к классу решающих задачу интерпретации данных и прогнозирования в условиях с квазидинамическими исходными данных и знаниями.

KPI системы – коэффициент прироста числа собираемых пользователями съедобных грибов.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧ

1.1 Обзор классификации грибов

Грибы, как отдельное биологическое царство, имеют обширную таксономию и внутреннюю классификацию (Рисунок 1.1):. Большое разнообразие видов делает нетривиальной задачу определения вида и съедобности найденного гриба.

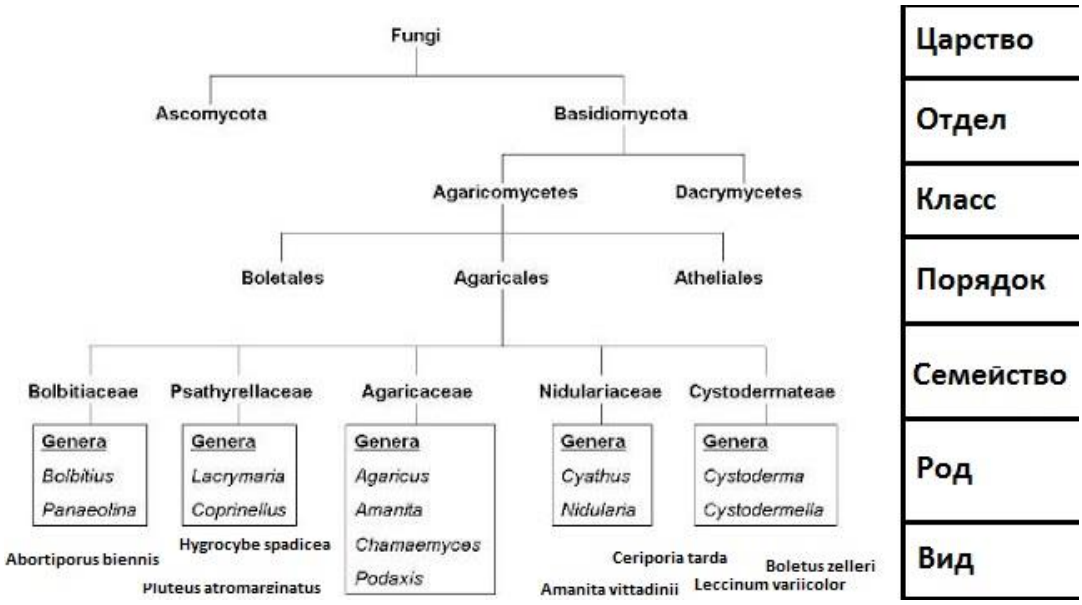


Рисунок 1.1 - Классификация царства грибов

Для человека съедобность гриба является его ключевой характеристикой. Несъедобный гриб может не только быть невкусным и вызывающим пищевое расстройство, но и смертельно ядовитым. Поэтому корректное и точное определение конкретного вида является основной проблемой при сборе грибов.

При разделении грибов на подгруппы, у них проявляются определенные объединяющие и разделяющие их виды таксономические признаки, по которым их можно классифицировать.

В целом человеческий мозг способен эффективно классифицировать разные предметы по внешнему виду и по другой доступной информации, в том числе и грибы, если человек опытен и подготовлен.

Основываясь на специализированных статьях[1][2][3] были определены следующие важные таксономические признаки:

- Морфологические признаки: форма ножки и шляпки; поверхность шляпки и гимениальная пластинка; цвет гриба;
- Место роста: почва, корни деревьев, деревья. Некоторые виды растут только на определенных породах деревьев;

– Географическое расположение, поскольку некоторые виды растут строго в определенных природных зонах;
– Время года, влияющее на: осадки, влажность, температуру, уровень света;
– Запах и реакция на прикосновение также могут дать информацию о виде.

Таким образом, при классификации грибов необходимо принимать во внимание множество факторов.

1.2 Распознавание образов с помощью искусственного интеллекта

Самым доступным способом передать информацию о грибах для пользователя является загрузка её фотографии, исходя из этого основным способом классификации для системы будет реализация модели распознавания визуальных образов.

На текущий момент почти все такие модели созданы на базе нейронных сетей и методов глубокого обучения. В начале 2010-х годов глубинные нейронные сети стали достигать огромных успехов в глобальных соревнованиях по машинному обучению, и к 2015 году они достижения смогли достичь человеческого уровня распознавания образов. Революцию в этой области произвели комплексные сети класса CNN (свёрточные нейронные сети), которые стали основной технологией для большинства будущих моделей, решающих задачи компьютерного зрения, в том числе проблемы многоклассовой классификации.

Исходя из этого было принято решение разрабатывать систему с моделью на базе CNN. Модели этого класса обучаются с учителем, и включают много возможностей для настройки параметров под предметную область и доступные ресурсы.

1.3 Постановка задачи

Итого, для реализации системы были поставлены следующие задачи:

1) Так как модели из области deep learning требуют большого времени обучения, было важно изучить существующие технологии и инструменты, и определить для работы наиболее подходящие и эффективные из них;
2) Нужно было собрать данные для обучения модели, причём учитывая разнообразие видов это должен был быть достаточно объёмный набор данных, поэтому сбор должен был быть автоматизирован;

3) Нужно было обработать и проанализировать собранные данные, чтобы подготовить их и саму модель к обучению;
4) Нужно было спроектировать и обучить модель классификатора исходя из особенностей предметной области и доступных данных;
5) Нужно было создать интерфейс для ввода новых образцов и вывода результата их классификации.

2 ИНСТРУМЕНТЫ И ТЕХНОЛОГИИ

2.1 Среда разработки

В качестве среды разработки был выбран сервис Colaboratory, или коротко "Colab", разработанный и поддерживаемый компанией Google. Язык программирования – Python.

Colab работает на основе Jupyter Notebook и удаленно запускает код языка Python в облаке. Доступ к Colab осуществляется через веб-фронтенд в любом браузере и не требует никаких дополнительных установок.

Выбор Colab обусловлен его архитектурой, в которой предустановлены и поддерживаются большинство Python библиотек для машинного обучения и анализа данных. Также данная среда позволяет выполнять код на высокопроизводительных серверах Google, в том числе и на модулях GPU. (GPU - англ. graphics processing unit, рус. графический процессор)

Выполнение на графическом процессоре особенно важно в случае решения задач машинного обучения. GPU показывают крайне высокую эффективность при реализации алгоритмов ML и DL, что значительно ускорило повторные процесс обучения сети при перенастройках параметров.

GPU обеспечивают высокую эффективность за счёт большого числа количества логических ядер, которые крайне производительны в параллельной работе с большим количеством однотипных несложных вычислений, которые преобладают в процессе обучение нейронных сетей. Например, вычисление сумм при активации нейронов, или вычисление градиентов, но особенно это полезно в операциях свертки в свёрточных нейронных сетях.

Также вычисления в глубоком обучении требуют обработки огромных объемов данных, и пропускная способность видеопамяти GPU позволяет укорить этот процесс.

2.2 Инструменты для веб-парсинга и работы с данными

Для веб-парсинга использовались популярные Python-библиотеки requests и BeautifulSoup:

- requests позволяет отправлять HTTP-запросы по url;
- beatifulsoup через подключение по HTTP анализирует HTML и XML документы, из текста которых извлекаются данные с их веб-страниц.

Для анализа и предобработки полученных данных использовалась библиотека pandas.

2.3 Инструменты для машинного обучения

Для построения и обучения нейронной сети использовался фреймворк TensorFlow и API для него – Keras.

TensorFlow также принадлежит компании Google и используется ей самой для работы над задачами распознавания, классификации и визуального обнаружения.

TensorFlow и Keras предустановлены в Colab и сопровождаются всеми повышающими эффективность работы надстройками, как аппаратными, так и программными.

3 СБОР ДАТАСЕТА

3.1 Общая информация

Для того, чтобы классификатор лучше определял загружаемые пользователями образцы, помимо высокого качества распознавания визуальных признаков, он также может учитывать другие особенности видов, например регион распространения.

Исходя из этого было принято решение собирать данные только по тем видам, что можно встретить в Казахстане. Таким образом, возможности классификации были улучшены путём исключения визуально схожих видов, но растущих только в определенных частях планеты.

Для большей унификации классов датасет был поделен на сабсеты по каждому региона.

По итогам поиска в Интернете найти датасеты изображений грибов только тех видов, что распространены в Казахстане, не удалось. Поэтому датасет потребовалось собирать вручную. Для этого были поставлены следующие задачи дата-инжиниринга:

1) сбор данных о распространенных видах грибов в Казахстане;
2) сбор изображений, чистка и структуризация;
3) анализ полученных данных ;
4) подготовка датасета под нужды системы, главным образом для реализации алгоритмов машинного обучения.

Финальный результат включает наборы изображений и метаданные для 455 произрастающих в Казахстане видах биологического царства «грибы». Всего было собрано и структурировано около ~85700 изображений. Это уникальный набор данных для Казахстана, аналога которому нет в сети, и он может служить основой для обучения других моделей, решающих задачу классификации.

Помимо этого датасет может быть использован для анализа в других сферах или для работы над какими-либо прочими проектами, где будет востребован.

Ссылка на страницу датасета в github:
https://github.com/forcelumerence/KZ_mushrooms_dataset

3.2 Парсинг метаданных

Для того, чтобы собрать датасет, было необходимо получить данные о видах Казахстана. В качестве источника этих данных был использован сайт

«Грибы Казахстана» fungi.su.

Процесс парсинга состоит из изучения HTML-кода страниц и извлечения текстовых данных из HTML или XML тегов на них.

Для получения HTML-документов использовалась библиотека requests, а для изъятия разметки из них в виде текста библиотека BeautifulSoup.

Для сбора основных данных по всем видам парсились страницы списка видов по ссылкам следующего образца:
http://fungi.su/infusions/advanced_articles_sort/mycota_reg.php?&rowstart=%d

Для того, чтобы переходить по страницам списка видов, которых всего было 16, был осуществлен цикл, который подставлял в ссылку значение, отвечающее за её порядковый номер в списке.

Для того, чтобы изъять нужные данные, нужно было определить html-теги, в которых они находились. Для этого была изучена разметка и HTML-код всех страниц, с которых собирались данные.

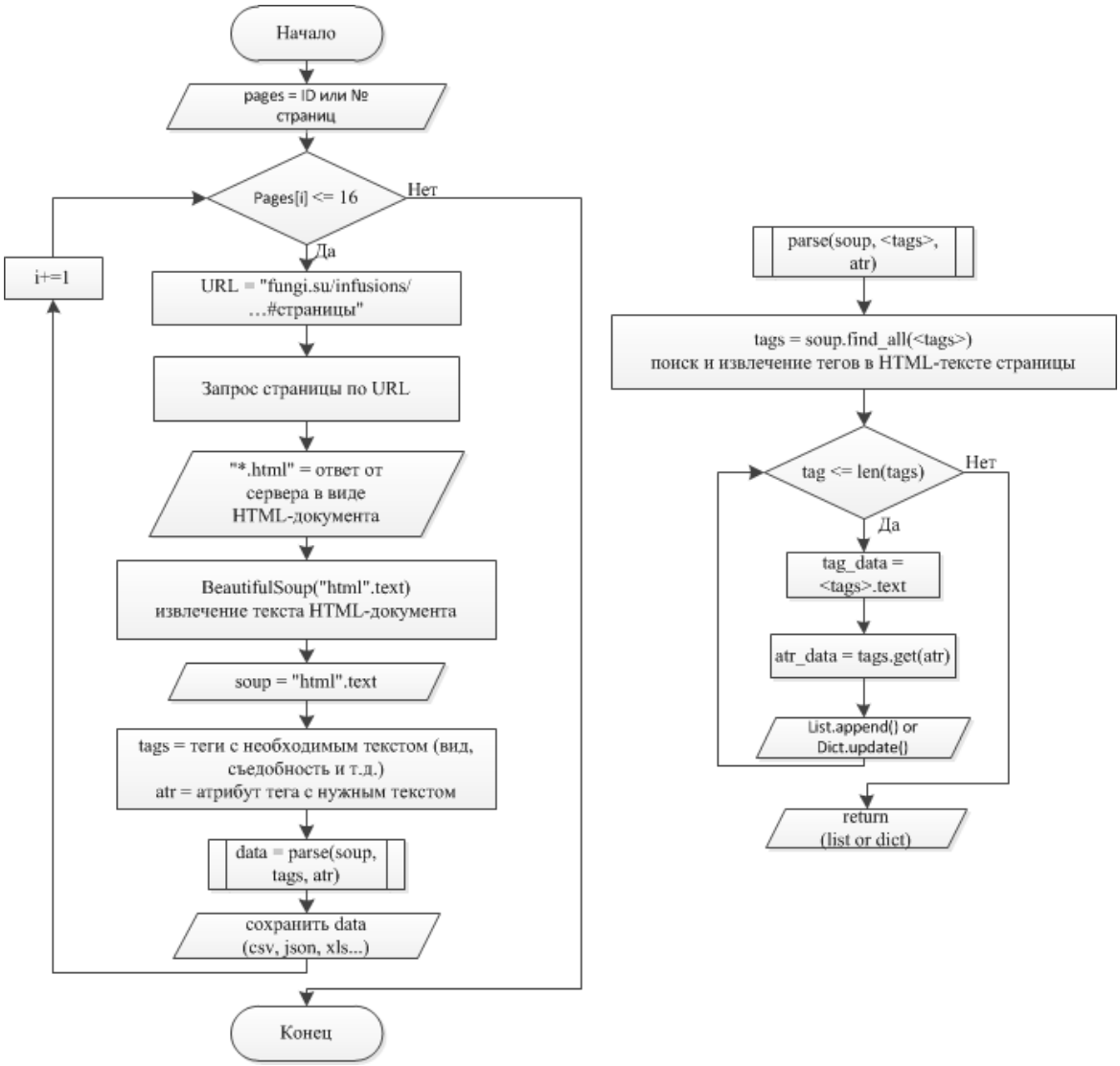


Рисунок 3.1 – Блок-схема алгоритма парсинга веб-страниц fungi.su

Собранные данные сохранялись в отдельные списки, и после окончания парсинга всех страниц были объедены в единый pandas-датафрейм (Рисунок 3.2):

	title_rus	title_eng	edibility	divisio	classis	ordo	familia	genus	location	article_id
0	Агроцибе ранняя	Agrocybe praecox	Съедобен	Basidiomycota	Agaricomycetes	Agaricales	Strophariaceae	Agrocybe	ЮК	2162
1	Альбатреллус тянь-шаньский	Albatrellus tianschanicus	Съедобен	Basidiomycota	Agaricomycetes	Russulales	Albatrellaceae	Albatrellus	ЮК	835
2	Антродия рядовая	Antrodia serialis	Несъедобен	Basidiomycota	Agaricomycetes	Polyporales	Fomitopsidaceae	Antrodia	ВК	2384
3	Аскоболус навозный	Ascobolus stercorarius	Несъедобен	Ascomycota	Pezizomycetes	Pezizales	Pyronemataceae	Ascobolus	ЮК	834
4	Аскокорине блюдцевидная	Ascocoryne cylichnium	Несъедобен	Ascomycota	Leotiomycetes	Helotiales	Helotiaceae	Ascocoryne	ЮК	2146
...	...	...	...	...	...	...	...	...	...	...
450	Энтолома весенняя	Entoloma vernum	Ядовит	Basidiomycota	Agaricomycetes	Agaricales	Entolomataceae	Entoloma	ЮК	661
451	Энтолома гладконогая	Entoloma poliopus	Несъедобен	Basidiomycota	Agaricomycetes	Agaricales	Entolomataceae	Entoloma	ЮК	1184
452	Энтолома садовая	Entoloma clypeatum	Съедобен	Basidiomycota	Agaricomycetes	Agaricales	Entolomataceae	Entoloma	ЮК	53
453	Энтолома светло-коричневая	Entoloma saepium	Съедобен	Basidiomycota	Agaricomycetes	Agaricales	Entolomataceae	Entoloma	ЮК	54
454	Энтолома шершавоножковая	Entoloma hirtipes	Несъедобен	Basidiomycota	Agaricomycetes	Agaricales	Entolomataceae	Entoloma	ЮК	756

455 rows × 10 columns

Рисунок 3.2 – Датафрейм с метаданными датасета

3.3 Загрузка, очистка и структуризация изображений

Для того, чтобы загрузить изображения с сервера, нужно спарсить прямые ссылки на них.

На схеме ниже (Рисунок 3.3) отображен путь до изображений по карте сайта. Было необходимо сделать суммарно переход по 4-м страницам и собрать ссылки на файлы изображений на сервере.

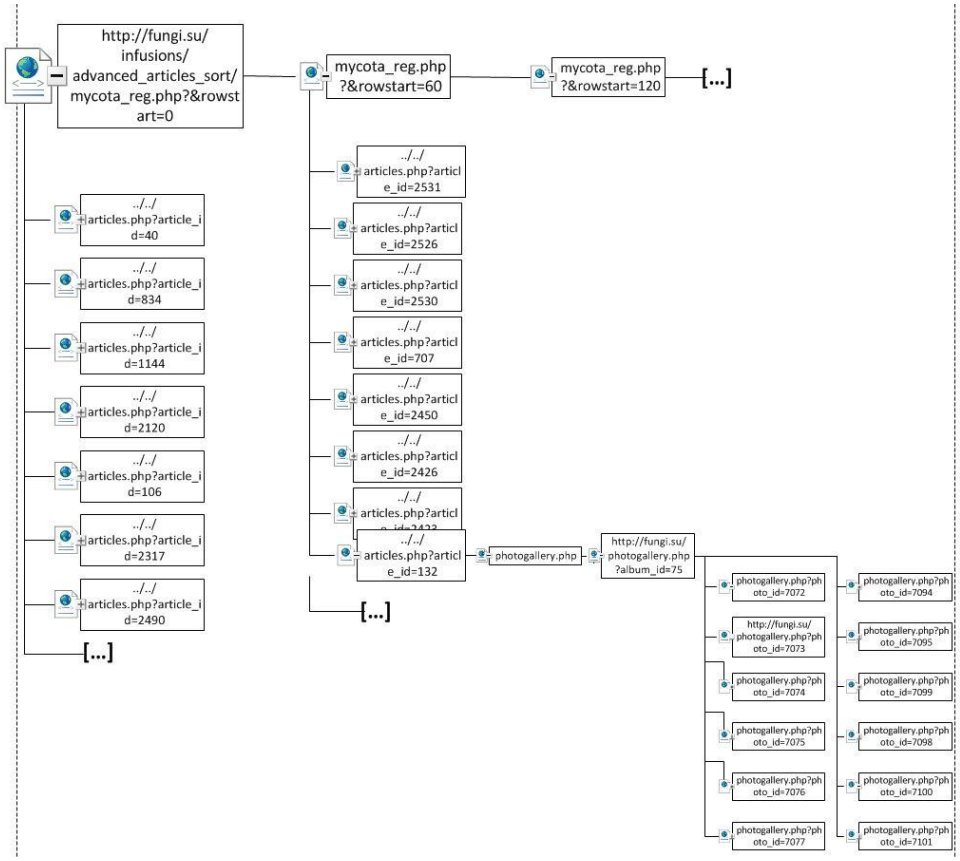


Рисунок 3.3 – Путь по карте сайта до прямых ссылок на фотографии

Все ссылки были записаны в словарь формата {'Название вида' : (список ссылок)} и сохранены в json файле.

После этого по словарю проходился цикл, который осуществлял запросы на скачивание файлов по ссылкам и сохранял их в директории, отдельные для каждого вида.

В результате этого было загружено около 7000 фотографий, и этого было критически недостаточно для обучения модели, поэтому работа по сбору данных было продолжена с использованием других источников.

Большая часть изображений была вручную выгружена из других датасетов путём поиска по названию. В случае слишком малого образцов какого-либо дополнительные фотографии собирались из Google и Яндекс картинок.

При сборе датасета было учтено то, что некоторые виды имеют несколько используемых названий (прим. Mycena strobilicola и Mycéna plúmipes, Coprinus domesticus и Coprinellus domesticus, и пр.). Если не удавалось найти достаточный набор изображений по одному названию, осуществлялась проверка по остальным.

После того, как было загружено достаточно число изображений всех видов, нужно было провести очистку данных от ложных образцов. Датасет был частично очищен от некорректных изображений, например фотографий спор под микроскопом, мицелиев, схем из учебников и т.п.

В очистки также удалялись виды, которые не похожи на привычные собираемые грибы, и которые не подвергались бы распознаванию пользователями, поэтому затраты ресурсов модели на них не целесообразны. Например, под такую фильтрацию попали некоторые плесневые грибы и грибы-паразиты.

Итоговый датасет был разбит по регионам, которые представлены в виде отдельных сабсетов. Структура датасета (Рисунок 3.3):

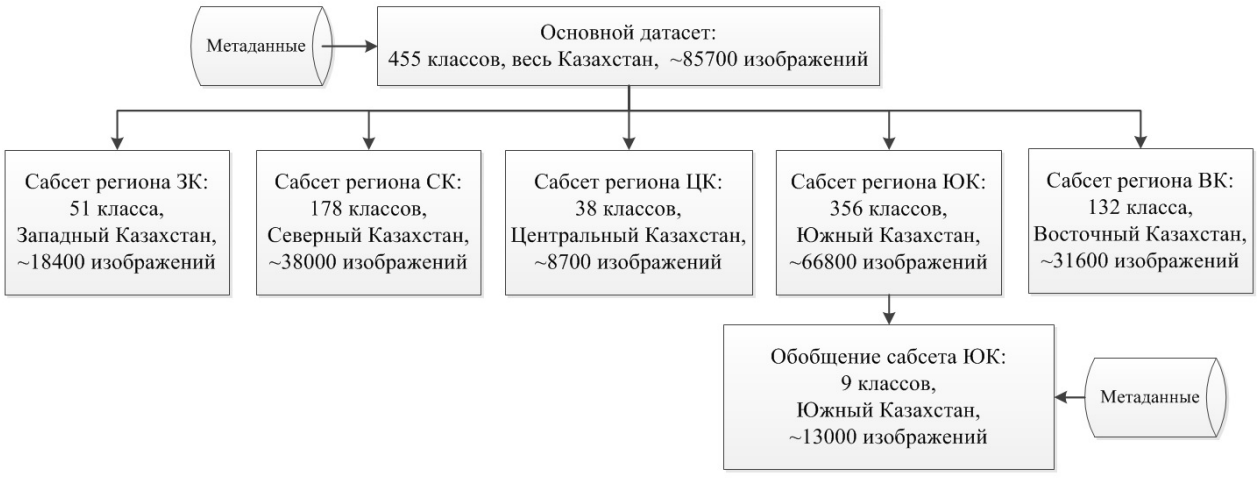


Рисунок 3.3 – Структура датасета

3.4 Анализ полученных данных и корректировки датасета

Классификация 455 видов - слишком обширная и сложная задача, для решения которой собранного числа данных совсем недостаточно большого объема данных было недостаточно. Чтобы сократить число классов, были предприняты следующие модификации датасета:

- Разбиение на сабсеты по 5 регионам Казахстана, включающие все произрастающие в них виды
- Обобщение видов до родов и семейств с учетом статуса съедобности и схожести морфологических признаков
- Обобщение до наиболее известных и встречаемых видов, поэтому например те грибы, которые невозможно собрать, были исключены

Для обучения был выбран сабсет Южного Казахстана, поэтому к нему были применены описанные выше правила, и в результате всех компромиссов итоговый список видов данного региона был обобщен до 9 преобладающих и наиболее схожих классов, каждому из которых был присвоен однозначный статус съедобности. Сабсет был пересобран в соответствии с этим классами, сократившись до суммарного числа в ~12500-13000 изображений.

Таблица 3.1 – Итоговый перечень классов обобщенного сабсета ЮК

Название (англ.)	Съедобность	Название (рус.)	Объем
Agaricus	Съедобен	Шампиньон	1653
Amanita	Несъедобен	Мухомор	1540
Boletus	Съедобен	Боровик	1473
Cortinarius	Несъедобен	Паутинник	1336
Entoloma	Несъедобен	Энтолома	1464
Hygrocybe	Несъедобен	Гигроцибе	1316
Lactarius	Съедобен	Груздь	1563
Russula	Съедобен	Сыроежка	1448
Suillus	Съедобен	Маслёнок	1362

В перспективе в случае получения данных о месяцах роста каждого из видов, система может быть улучшена с помощью дополнительного разбиения датасета по этому критерию. Это позволит снизить обобщение по родам и семействам, повысив качество и информативность предсказаний классов.

Пополнение базы изображений также очевидно улучшит качество датасета.

4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

4.1 Архитектура экспертной системы

Архитектура экспертной системы (Рисунок 4.1) включает:

- Эксперта (разработчик) и среду разработки;
- Базу знаний (собранные обучающие данные);
- Механизм решений;
- Интерфейс взаимодействия с пользователем.

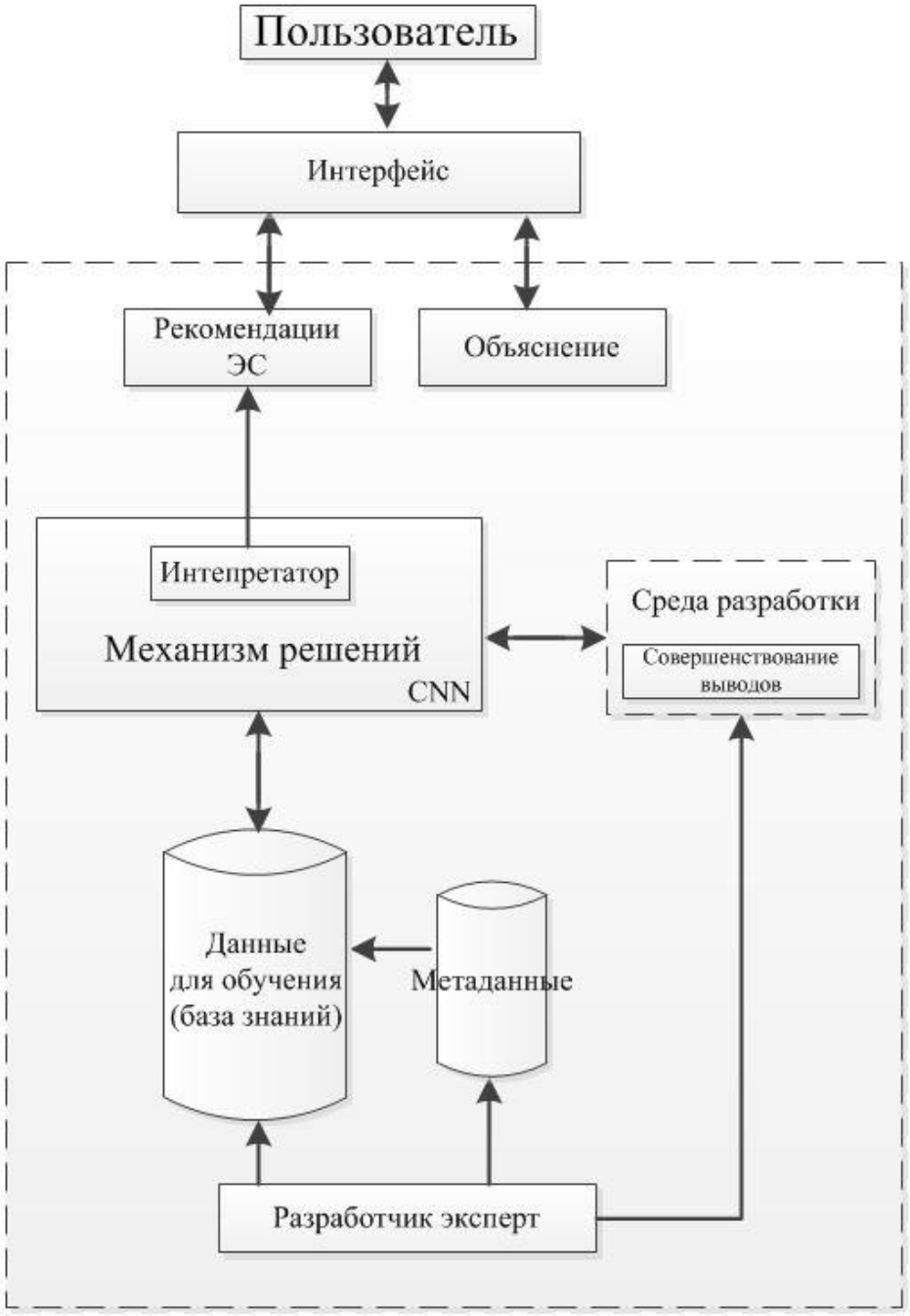


Рисунок 4.1 – Архитектура экспертной системы

В качестве механизма решений была спроектирована и модель ИИ-классификатора с архитектурой свёрточных нейронных сетей, после чего она была обучена на базе собранных данных.

Пользователь передаёт путь на файл с изображением, классификатор делает интерпретирует его и делает предсказание, после чего выводит результат распознавания пользователю, который сопровождается дополнительной информацией, а именно: вероятности того, какой класс представлен на фото, название на английском и русском языках, и данные о съедобности видов.

Исходя из полученной информации пользователь может принять решение о том, собирать найденный образец или нет.

Для база знаний использовался собранный специально для проекта этой системы уникальный датасет грибов Казахстана, а также метаданные о нём.

4.2 Модель классификатора

Задача многоклассовой классификации изображений с большим разнообразием обычно решается с применением методов глубокого обучения моделей (deep learning).

Глубокое обучение - это область машинного обучения, целью которой является изучение сложных функций, принимающих во внимание большое число параметров. Методы этой области требуют больших вычислительных и временных ресурсов на обучение, поэтому процесс обучения просчитывался в облаке на серверах Google с помощью Colab.

В качестве классификатора для решения поставленной задачи была выбрана модель глубинной нейронной сети – математической модели, которая построена по образцу человеческого мозга, а глубинной она называется из-за архитектуры с большим числом скрытых слоев.

Если человеку взять определенный гриб и попробовать визуально определить его вид, то глаза сразу будут выделять определенные признаки: какой у него цвет, форма шляпки, толщина ножки и т.д. После мозг соотнесет их с имеющимися знаниями о видах, чтобы определить класс наблюдаемого образца.

Нейронная сеть работает схожим образом. Она сначала выделяет признаки и после в соответствии с заложенными в неё весами этих признаков определяет по их сумме, какой это вид.

Например, веса мухомора можно представить как:

- толстая ножка 0.2

– красная шляпка 0.2
– точки на шляпке 0.6

При наличии одних только точек на шляпке общий вес будет 0.6, что превышает сумму первых двух особенности. Даже если попадётся гриб с тонкой ножкой, и шляпка будет не красной, но в крапинку, то этот гриб будет определен нейросетью как мухомор.

Конечная цель обучения классификатора – определить генеральную совокупность ключевых признаков отдельных классов и высчитать их весовые коэффициенты, чтобы уже по ним классифицировать любые подаваемые изображения.

Для обучения нейронной сети нужен «учитель» – набор размеченных данных, чтобы сеть могла найти и соотнести между собой признаки изображений одного класса.

Разработанная модель относится к особому классу нейронных сетей, которые называют CNN, или же сверточными нейронными сетями. Название происходит от особых скрытых слоев в этой сети, которые проводят над данными операцию свёртки.

CNN состоит из четырех основных частей:

1) Входной слой;
2) CNN-слои свертки и сжатия, выделяющие ключевые признаки;
3) Полносвязные слои, корректирующие весовые коэффициенты признаков;
4) Выходной слой.

На входной подаётся изображение и преобразуется в математический вид, после чего на сверточных слоях к нему начинают применяться операции свёртки.

Операция свёртки накладывает на изображение фильтр заданной размерности, который последовательно с заданным шагом двигается по всей площади матрицы и накладывает специальные фильтры свёртки.

Фильтр поэлементно умножается на соответствующие элементы в позиции наложения на канал изображения, полученные значения складываются. Сумма переносится на следующий слой в позицию, где был центр фильтра свёртки. Результат полной свёртки изображения одним фильтром называется картой признаков.

Структура CNN адаптирована для анализа пространственных особенностей и признаков изображений. Каждый свёрточный слой выполняет свою функцию по обобщению признаков выделяемых фильтрами, а каждый фильтр в свою очередь получает свою специализацию и анализирует на

изображении конкретные каналы (геометрические, цветовые). Например, края и углы отдельных частей предмета на изображении, точки или цветовые кластеры, прямые линии и изгибы и т.д.

Фильтры инициализируются со случайными значениями и автоматически корректируют свои значения по заданному алгоритму оптимизации.

Пример наложения разных фильтров свёртки (Рисунок 4.2)

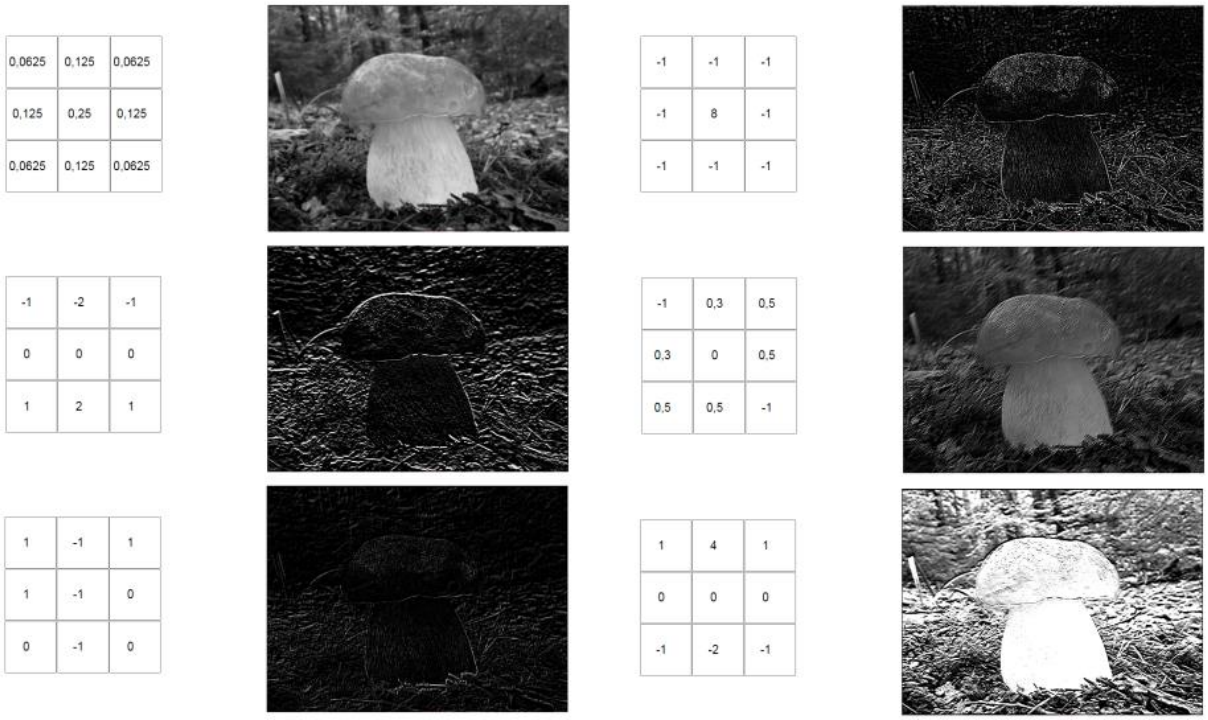


Рисунок 4.2 – пример наложения разных фильтров свёртки на фото

После свёрточных слоев располагаются слои сжатия, которые сжимают переданные им карты признаков, уменьшая их размер и производя обобщение признаков для поиска закономерностей.

Идея состоит в том, что выявленные свёрткой признаки можно немного объединить, потому что для дальнейшей обработки их подробное отображение уже не требуется. К тому же фильтрация уже ненужных деталей помогает не переобучаться.

После CNN-слоев данные, описывающие наиболее выявленные абстрактные и общие признаки изображений, объединяются в одномерный вектор и переходят в полносвязный слой, где проходят пересчёт весов и выводятся в выходной слой, показывающий вероятность принадлежности изображения к какому-либо классу.

Пример движения изображения по CNN (Рисунок 4.3):

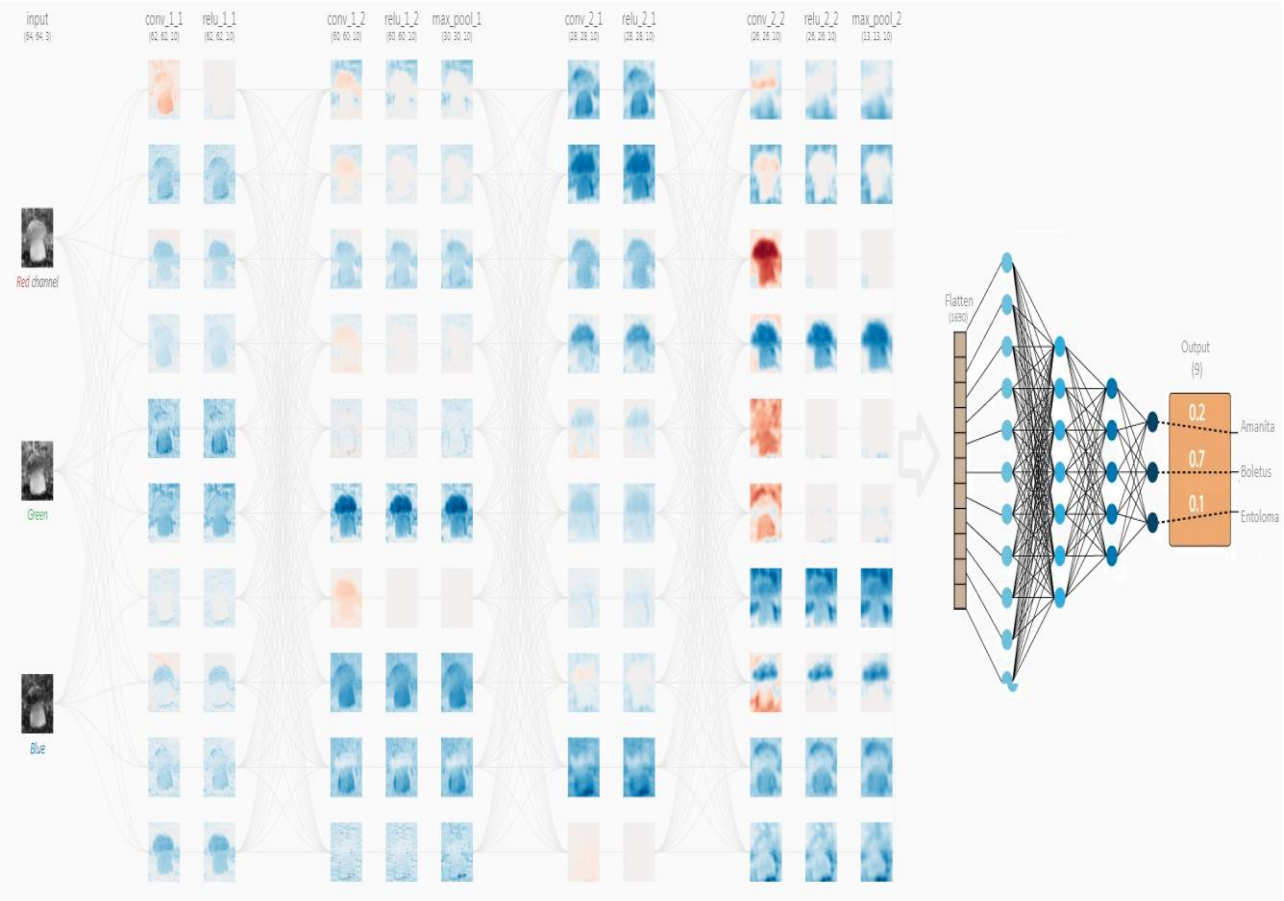


Рисунок 4.3 – Пример движения изображения по сети

Перед обучением модели необходимо настроить, а после и множество раз корректировать, её гиперпараметры, которые определяют в том числе и архитектуру сети. Также важно предобработать датасет и подготовить к передаче в нейросеть. В результате такого комплексного подхода модель может достигнуть высокой точности классификации.

Общий алгоритм работы модели (Рисунок 4.3):

Рисунок 4.4 – Общий алгоритм работы модели

4.3 Гиперпараметры модели

Гиперпараметры – это параметры для управления процессом обучения, значения которых устанавливаются перед запуском этого процесса.

4.3.1 Число слоев

Помимо обязательных входного и выходного слоя, основой настройки модели будет определение скрытых слоев свёрточной и полносвязной нейросетей. Число промежуточных слоев играет крайне важную роль, оно влияет на скорость и качество обучения, на проявление переобучения.

4.3.2 Число нейронов в слоях

На входном слое это число будет равно количеству элементов входных данных. Например, одно изображение размером 240x240 пикселей будет подавать на вход один 3D тензор размерности (240, 240, 3) или три 2D матрицы размерности [240, 240], которые заполнены значениями цветового диапазона пикселей каждого цветового канала RGB (красный, зеленый, синий).

Число нейронов выходного слоя будет равно 9, что равняется числу классов, присвоенных изображениям датасета.

4.3.3 Параметры свёрточных слоев

В Keras свёрточные слои для изображений определены классом Conv2D (filters, kernel_size, strides=(1,), …). Conv1D используется для аудио, а Conv3D для видео.

Основные параметры этого класса:

- Filters - число фильтров свёртки;
- kernel_size - размер фильтра;
- strides - шаг сдвига фильтра свёртки по плоскости матрицы/тензора (по умолчанию - 1);
- padding - определяет размер матрицы после свёртки, при значении True карта признаков без сохранять размер изначальной матрицы, иначе уменьшится на 2 строки и 2 столбца.

Число фильтров влияет на количество выискиваемых признаков. При слишком большом количестве фильтров, модель может начать принимать случайные совпадения за признаки, при слишком маленьком – упускать многие детали.

Размер фильтра определяет то, какого вида особенности будут выделяться из изображения. Чем меньше этот размер, тем более мелкие и локальные признаки выделяются. Значения фильтров заполняются случайно, но их также можно загрузить из предобученной модели, что и было реализовано в итоговой модели.

4.3.4 Параметры слоев сжатия

В Keras слои сжатия определены классом *Pooling2D(pool_size=(2,2), strides=2, padding='valid'), где на месте * стоит критерий сжатия.

Параметры этого класса:

- если критерий сжатия Max, то из окна сжатия будет браться максимальное значение, если Avg – то среднее между всеми;
- pool_size - размер окна, из которого выбирается значение;

– stride – шаг сдвига по плоскости матрицы (по умолчанию – 2).

От размера окна будет зависеть то, насколько сильно уменьшится изображение. Редко изображение уменьшают более, чем в 4 раза, в этом случае размер окна будет (4, 4). Чаще сжатие производится с окном (2, 2), т.е. уменьшением в 2 раза.

4.3.5 Критерий качества обучения (loss-функция, функция потерь)

Критерий качества (loss-функция) определяет функцию, по которой будут оцениваться полученные весовые коэффициенты. Чем меньше значение этой функции, тем выше качество подобранных весов.

В разработанной модели как критерий качества использовалась функция категориальной перекрестной энтропии, которая сравнивает распределение результатов, которые получила модель, и верные ответы. Это основная функция, применяемая в задачах многоклассовой классификации.

Процесс обучения и точность классификации зависят от того, насколько хорошо получилось минимизировать значения функции потерь. Процесс минимизации производится алгоритмом градиентного спуска, с применением к нему различных алгоритмов оптимизации.

4.3.6 Алгоритмы оптимизации градиентного спуска и их параметры

Задача алгоритмов оптимизации – увеличить скорость сходимости градиента к точке глобального минимума функции потерь. Сам по себе процесс минимизации обычным градиентным спуском занимал бы огромное количество времени, при этом без гарантии, что он градиент в итоге сойдётся.

В процессе оптимизации важно прийти или приблизиться именно к глобальному минимуму, и избежать локальный минимум.

Learning rate (LR), или же шаг сходимости – это коэффициент скорости оптимизации, который определяет значение шага в правильном направлении для минимизации loss.

Чем LR больше, тем быстрее происходит сходимость алгоритма, однако в этом случае выше шанс «перепрыгнуть» глобальный минимум. При этом уменьшение LR поможет улучшить точность поиска минимума, но может слишком сильно замедлить процесс. Соответственно, когда в модели тысячи весовых коэффициентов проходят минимизацию, изменение LR серьёзно влияет на скорость обучения.

В глубоком обучении вместо стандартного алгоритма обратного распространения ошибки используется более эффективные оптимизаторы,

например алгоритмы Adam (Adaptive Moment Estimation) и стохастический градиентный спуск (SGD).

Adam является одним из наиболее производительных алгоритмов оптимизации градиентного спуска в машинном обучении, демонстрируя высокую вычислительную эффективность в задачах с большим объёмом данных. Преимущества Adam перед другими алгоритмами:

1) свойство momentum позволяет создать схожий физическому импульс при движении градиента в одном и том же направлении, ускоряя таким образом сходимость;
2) Adam способен адаптировать скорость обучения, автоматически повышая значение learning rate, тем самым замедляя обновление градиентов для признаков, которые уже показывают быструю сходимость, что, например, часто происходит на начальных слоях.

Другой высокоэффективный алгоритм оптимизации – это стохастический градиентный спуск (SGD) по Нестерову, в котором также как и в Adam используется momentum (Nesterov momentum), но при этом нет возможности адаптировать learning rate

SGD по Нестерову и Adam – основные оптимизатора, используемые при обучении глубинных нейронных сетей.

Стартовым оптимизатором обычно выбирается Adam из-за своей возможности корректировать скорость сходимости отдельных параметров, и также у него хорошо откалиброваны гиперпараметры, и они не требуют дополнительной настройки, что крайне полезно и удобно в начале обучения.

SGD чаще используется, когда веса и некоторые параметры модели уже настроены после запусков обучения с Adam.

4.3.7 Функции активации

Функция активации определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения. Она может быть отдельно настроена для скрытых и выходного слоев.

Основная применяемая в глубоком обучении функция активации – это функция ReLU, которая выделяет положительную часть своего аргумента:

$$f(x) = \max(0, x)$$

т.е. при положительном аргументе она вернет сам аргумент, а при отрицательном – 0. Это свойство позволяет нейронной сети корректировать весовые коэффициенты в равной степени на всех слоях, в отличие от функций, в которых маленькое значение аргумента на некоторых слоях сети может существенно снизить веса.

ReLU была определена функцией активации на всех скрытых слоях модели, но выходе для этого была выбрана softmax.

Softmax — это обобщение логистической функции для многомерного случая. Функция softmax возвращает значения в интервале [0, 1] и их сумма будет составлять 1. На выходном слое сети эти значения можно интерпретировать как вероятности принадлежности внесенного наблюдения к разным классам, что крайне полезно в задаче классификации.

С этой функцией можно будет продемонстрировать, к каким видам грибов может относиться внесенный образец, что позволит пользователям проще понимать, с чем сравнивать найденный образец, что подтвердить прогноз класса.

4.4 Оптимизация гиперпараметров

Единственный способ найти подходящую схему сети - это подбирать их методом множества проб и ошибок. Нет какого-либо способа достоверно вычислить оптимальное число слоев или другие гиперпараметры.

Настройка гиперпараметров – непрерывный итеративный процесс, который сильно зависит от предметной области, подаваемых данных и поставленной задачи. Наилучшие параметры достигаются подгонкой архитектуры модели под эти аспекты.

Чтобы эффективнее подбирать параметры, стоит выдвигать и проверять различные гипотезы. Например, предположение о том, что для видов грибов требуются большое число сверточных слоёв и фильтров свёртки в них, чтобы определять более сложные детализованные признаки и связи между ними. Соответственно можно проводить тесты модели, начиная с малого числа слоев и постепенно их наращивая.

Одним из наиболее эффективных способов выполнения тестов, чтобы проверять гипотезы и перенастраивать параметры, является проверка качества обучения на отдельном наборе данных, который называется выборкой валидации.

Валидация – это проверка точности предсказаний модели, которая производится на неиспользовавшемся в обучении независимом наборе данных. Точность оценивается каждую итерацию, в результате после окончания обучения можно отследить, в каком момент она начала падать или стагнировать.

Валидация позволяет также отследить момент, на которой началось расхождение показателя точности обучающей и валидационной выборки, что характеризует старт переобучения.

Если после изменений гиперпараметра, точность на валидационной выборке стала выше, то это изменение считается полезным и сохраняется, в ином случае процесс подбора и корректировки продолжается.

Ещё благодаря валидации возможна корректировка параметров прямо в процессе обучения и оптимизации с помощью специальных callback-функций, отслеживающих показатели валидационной выборки.

Например, при первых попытках обучения в метод fit была включена callback функция ReduceLROnPlateu(), которая в случае прекращения роста val_accuracy (точности в валидационной выборке) корректировала learning_rate, уменьшая или увеличивая его на один разряд после эпохи, на которой была зафиксирована стагнация. Однако это замедляло процесс сходимости оптимизатора, что приводило к серьёзному увеличению требуемого на обучение времени, поэтому данная функция была отключена, и внимание было переключено на другие гиперпараметры.

Другая callback-функция, EarlyStopping(), наоборот позволила ускорить процесс настройки и совершенствования модели. При стагнации val_accuracy эта функция замораживала весовые коэффициенты с последней эпохи и останавливала процесс обучения модели.

Последней включенной в процессе обучения функцией такого рода был метод ModelCheckpoint(), который в случае обновления минимума критерия качества (loss-функции) после эпохи сохранял веса с этой эпохи в .h5 файле.

В целом, процесс оптимизация гиперпараметров нужен как для улучшения качества модели, так и для ускорения обучения, которое при определенных настройках модели могло занимать целый день. Для решения этой проблемы помимо callback-функций в архитектуру модели нейронной сети также были добавлены слои, отвечающие за нормализацию.

Слои batch normalization приводят математическое ожидания и дисперсию в предыдущем слое к значениям 0 и 1, таким образом ограничивая распределение элементов батчей на предыдущем слое архитектуры и нормализуя их. Нормализация позволяет экономить время на адаптации сети между батчами.

4.5 Проблема переобучения

Переобучение – явление, когда модель находит и настраивает весовые коэффициенты под уникальную для обучающей выборки закономерность

признаков. В таком случае она будет показывать высокую точность классификации, но только на образцах из данной выборки.

При переобучении модель утрачивает способность к обобщению и выявлению генеральной совокупности признаков свойственных определенному классу, и избегание данной ситуации является важным аспектом проектирования.

Как было упомянуто ранее, отследить старт переобучения можно с помощью валидации. Если будет известен момент, когда началось переобучение, то можно весовые коэффициенты фиксируются на значениях перед этим, и в последствие модель начинает обучение с них, но уже с другими гиперпараметрами.

В данные были разбиты на три набора:

1) Обучающая выборка – набор данных, которые подаются в нейронную сеть и на основе которой происходит обучение и подбор весовых коэффициентов;
2) Валидационная выборка – оценочный набор, который задействуется непосредственно в процессе обучения и позволяет отслеживать переобучение модели. Каждую эпоху на образцах этого набора высчитывается значение loss-функции и точность классификации. Если эти показатели будут значительно расходиться с показателями обучающей выборки, то это будет говорить о переобучении;
3) Тестовая выборка – набор данных, на котором тестируется точность классификации уже обученной модели.

Также чтобы избежать переобучения была применена нормализация данных. Нормализация стандартизирует входные до диапазона от 0 до 1, также надо будет делать при эксплуатации нейронной сети путем разделения каждого входного элемента на максимально возможное значение в массивах, в случае цветовых диапазонов оно равно 255

4.6 Архитектура модели

4.6.1 Проектирование архитектуры

Итоговая архитектура модели (Рисунок 4.5) была спроектирована после проведения десятков экспериментов с подбором разных гиперпараметрови проверки разных предобученных сетей, пока не была получена удовлетворительная точность валидационной и тестовой выборок. Модель представляет из себя последовательную нейронную сеть с прямой связью

между всеми слоями.

На базе данной архитектуры была построена, скомпилирована и обучена финальная модель, весовые коэффициентов с которой реализуют классификацию в системе.

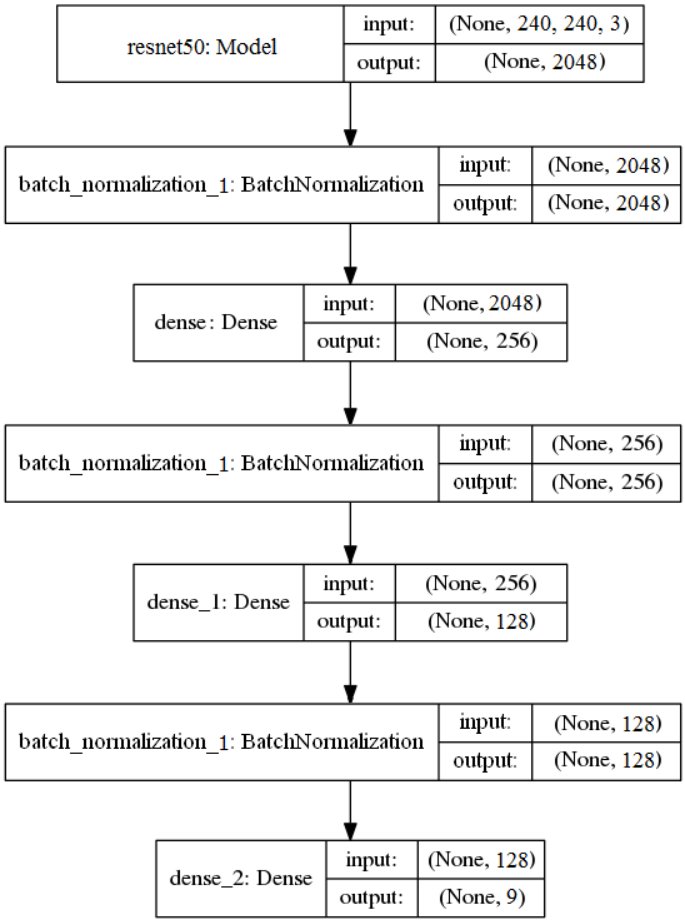


Рисунок 4.5 – Итоговая архитектура модели

4.6.2 Архитекутра CNN - ResNet50

Изначально сеть обучалась на оригинальных спроектированных CNN, но в результате было решено перепроектировать её с основой в виде предобученной сверточной нейронной сети.

В различные фреймворки для Deep Learning, в том числе и Tensorflow, включены пакеты с предварительно обученными моделями CNN с уже инициализированными стартовыми весами и настроенными гиперпараметрами. Многие из этих сетей имеют очень глубокую структуру и обучались на протяжении длительного времени, вплоть до нескольких дней.

Большие предобученные модели имеют большое преимущество в плане возможностей ускорить процесс настройки сети, потому что после

присоединении к архитектуре модели их весовые коэффициенты и гиперпараметры замораживаются и больше не корректируются.

Принимая во внимание эти особенности, было принято решение взять за основу архитектуры классификатора модель такого типа под названием ResNet50, что расшифровывается как Residual Network 50 - остаточная нейронная сеть с 50-ю слоями. Она была обучена на датасете ImageNet, объём которого составлял 1.2 миллиона изображений, разделенных на 1000 категорий.

Residual Learning переводится как «остаточное обучение», когда слои стремятся корректировать ошибки, допущенные слоями до него, вместо того чтобы просто изменять входы предыдущих слоев.

Архитектура сети поделена на условные уровни, которые отличаются числом слоев и фильтров в них (Рисунок 4.6):

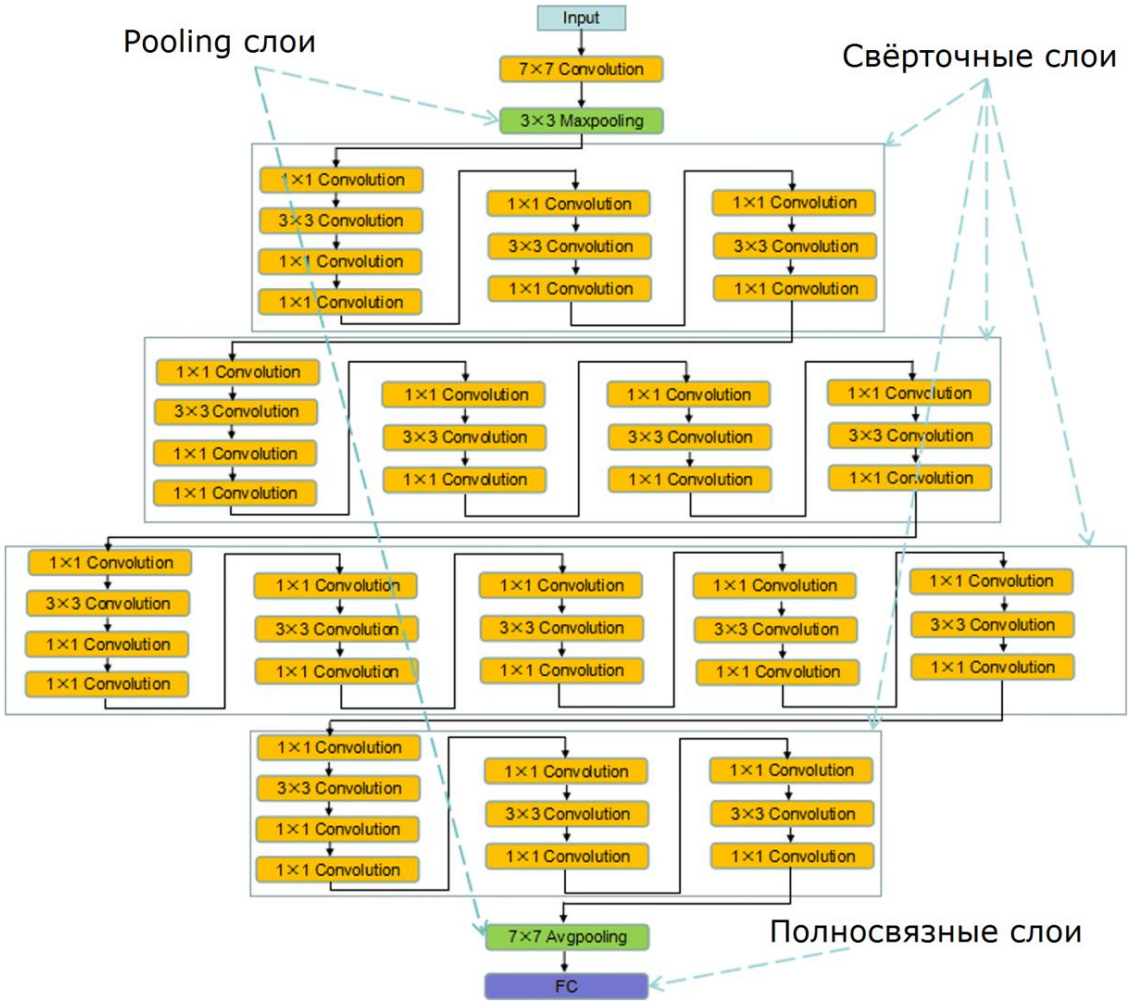


Рисунок 4.6 – Архитектура ResNet50

Число фильтров свёртки возрастает от уровня к уровню, начиная с 64 фильтров и заканчивая 2048. Размеры фильтров при этом также небольшие – 1x1 и 3x3 пикселя, с шагом в 2 пикселя. Такая сеть выделяет самые небольшие детали распознаваемых категорий.

При этом сжатие происходит только 2 раза: после первого свёрточного слоя (слой MaxPooling2D 3x3, уменьшение в изображения в 3 раза) и после последнего скрытого слоя (слой AveragePooling2D который переносит 4D тензоры в одномерный вектор и выводит его на полносвязную сеть).

Главным преимуществом такой архитектуры является большое количество слоев и фильтров, способных выделять наиболее уникальные признаки. Это могут быть совсем небольшие детали изображений, которые даже не встречаются на всех образцах класса, а также взаимосвязи таких деталей.

Для решаемой задачи такое свойство модели очень полезно. Среди всего видового разнообразия грибов существует множество ядовитых опасных видов-двойников, оригиналы которых безопасны и часто даже популярны.

Например, как и среди шампиньонов есть ядовитый вид шампиньон желтокожий, у которого ключевым отличием во внешнем облике от безопасных видов является небольшое желтое пятно на шляпке. Схожая ситуация с опятами и двумя его ядовитыми двойниками: ложноопёнок кирпично-красный и ложноопёнок серно-жёлтый. У обоих видов гимениальные пластинки отличаются насыщенным желтым цветом.

Именно потому, что при разборе изображений грибов очень важно выделять самые крохотные детали, и была выбрана сеть ResNet50.

При попытках обучить сеть до добавления модели ResNet50, число фильтров составляло 32, 64 и 64 (на 3-х сверточных слоях) с шагом 1 и размер 3x3. Той модели удалось достичь точности ≈60-65%, с ResNet это значение выросло до ≈82%.

4.6.3 Архитектура полносвязной сети

После сверточных слоев модель переносит данные в полносвязную сеть с 2 скрытыми слоями и 1 выходным слоем. На входной (выходной слой CNN) и на два скрытых слоев накладывается нормализация батчей, чтобы оптимизировать и ускорить адаптацию сети между итерациями.

Всего на вход в сеть приходит 2048 параметров, которые связываются с 256 нейронами последующего слоя и активируются функцией ReLU. Тоже самое происходит между скрытыми слоями, где связываются 256 и 128 нейронов, и с выходным слоем, где связываются 128 и 9 нейронов. Функция активации на выходном слое – softmax, подающая на выход вектор значений от 0 до 1, которые означают принадлежность к классу.

В качестве алгоритма оптимизации градиентного спуска был определен Adam с learning rate равным 0.001 и loss-функцией категориальной

перекрестной энтропии, которая важна для того, чтобы правильно работала функция softmax на выходе.

4.7 Предобработка данных

Для того, чтобы запустить обучение в скомпилированной модели, нужно загрузить данные из датасета и провести их предварительную обработку.

Все фотографии видов организованы по соответствующим директориям. Каждая директория будет соответствовать одному классу.

Как было сказано ранее, данные были поделены на 3 выборки: обучающую, валидационную и тестовую.

И в валидационную, и в тестовую выборку было отложено по 10% образцов датасета.

После этого необходимо было провести 3 операции:

1) Изменить размер всех изображений до 240x240 пикселей;
2) Разделить данные по батчам, чтобы пропускать их поочередно, а не все разом;
3) Наложить на данные аугментации.

Keras позволяет осуществить все 3 операции с помощью класса ImageDataGenerator. В параметрах конструктора данного класса задаются параметры аугментаций изображений.

Аугментации - это методы, которые позволяют размножить датасет путем добавления преобразованных или измененных копий уже существующих изображений. Примеры преобразований: повороты, смещения, отражение по оси, вращение, смена контрастности, гаммы и т.п. (Рисунок 4.5)

Рисунок 4.7 – Пример аугментаций

Аугментации позволяют значительно увеличить объем и разнообразие данных, при этом нейронная сеть будет считать каждое такое изображение уникальным, что всегда положительно сказывается на эффективности при обучении глубинных нейронных сетей. Главное, чтобы они сохраняли ключевые признаки образов.

Пример конфигурации ImageDataGenerator (Рисунок 4.6):

```
train_gen = ImageDataGenerator(
        rescale=1./255,                    #нормализация
        rotation_range=20,                 #поворот от -20 до +20 градусов
        width_shift_range=0.2,             #смещение изображение по ширине
        height_shift_range=0.15,           #смещение изображение по высоте
        shear_range=0.2,                   #растягивание изображения
        zoom_range=0.2,                    #увелечение/уменьшение
        brightness_range=(0.5, 1.0),       #яркость
        horizontal_flip=True,              #отзеркаливание по горизонтали
        )
```

Рисунок 4.8 – Пример конфигурации генерации аугментаций

Сконфигурированный объект ImageDataGenerator с использованием метода flow_from_directory(directory = `путь к директории выборки`) позволяет в реальном времени разделять данные по батчам заданного размера и генерировать аугментированные изображения на ходу каждую эпоху. За счёт этого достигается большее разнообразие данных, более уникальные изображения при обучении каждой эпохи позволяют избежать переобучения модели.

Также этот метод приводит все изображения, подаваемые в сеть, к одинаковому формату и размерности. В разработанной модели изображения трансформируются до разрешения 240x240 пикселей с 3 цветовыми каналами формата BGR.

Размер батча составляет 100 изображений на итерацию, итого на вход сети каждую итерацию подавался 4D тензор размерности (100, 240, 240, 3). При этом в нейросеть данные вносятся случайным образом, а не по порядку, что тоже помогает в устранении переобучение.

4.8 Процесс обучения

Обучение модели происходит по эпохам, которые распределены на число итераций, равное количеству батчей. На каждой итерации в сеть подаётся один

батч с изображениями, из которых извлекаются признаки, оптимизируется loss-функция и корректируются весовые коэффициенты.

Для обучения используется метод model.fit() (Рисунок 4.9):

Рисунок 4.9 – Алгоритм обучения методом model.fit()

В параметрах model.fit() указываются:

1. Ссылка на обучающую выборку;
2. Ссылка на валидационная выборку, на которой тестируются результаты каждой эпохи;
3. Число эпох;
4. Количество итераций на эпоху (совпадает с числом батчей);

5. Используемые callback-функции.

Описание алгоритма обучения разработанной модели CNN:

1) Из изображения операцией свёртки выделяются карты признаков и проходят операцию сжатия
2) элементы карты признаков переносятся в полносвязную нейронную сеть, образуя единый вектор признаков, который после связывается с нейронами скрытого слоя
3) Нейронная сеть высчитывает сумму весовых коэффициентов на связях между нейронами, пропускает её через функцию активации ReLU и получает значения весов для разных скомбинированных признаков
4) Нейрон выходного слоя с максимальным значением после активации softmax будет означать, что на изображении класс, соответствующий номеру этого нейрона
5) Полученные сложением признаков классы и реальные классы сравниваются, и если они расходятся, нейросеть корректирует веса признаков, с целью минимизировать перекрестную категориальную энтропию (loss-функцию)
6) Минимизация происходит методом градиентного спуска с применением алгоритма оптимизации Adam,
7) Сеть повторяет данный процесс для всех изображений обучающего набора по несколько раз, равных числу заданных эпох,
8) Как процесс обучения будет завершен, весовые коэффициенты будут заморожены и при передаче новых изображений в сеть, она сможет предсказывать их класс.

Итого при передаче на вход сети 4D тензора (100, 240, 240, 3) на выходе будет выведен вектор вероятностей (1, 9). Изображению присваивается класс с максимальным значениям в векторе.

Время обучения финальной модели заняло ~4 часа.

4.9 Оценка модели

Оценка модели производится с помощью Keras-метода model.evaluate() и истории валидации.

evaluate() выводит значения точности и loss-функции для валидационной и тестовой выборки:

- Validation Accuracy = 0.8218;
- Validation Loss = 0.6264;
- Testing Accuracy = 0.8262.

Далее приведены графики изменений значений точности (Рисунок 4.8) и критерия качества loss (Рисунок 4.9) обучающей и валидационной выборок по эпохам в процессе обучения:

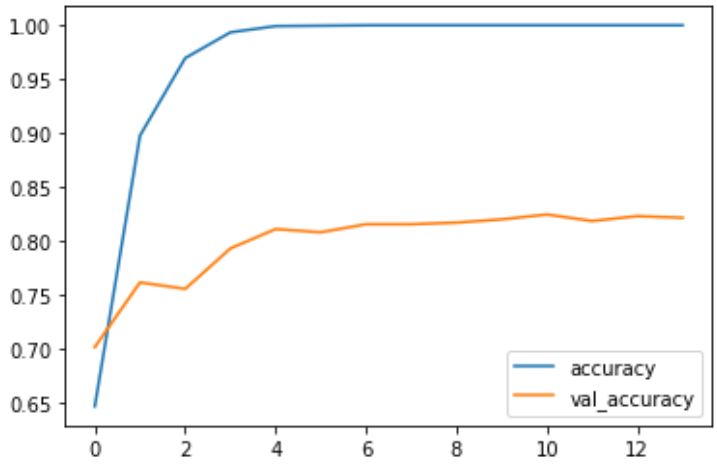


Рисунок 4.10 – График изменения точности с ходом эпох

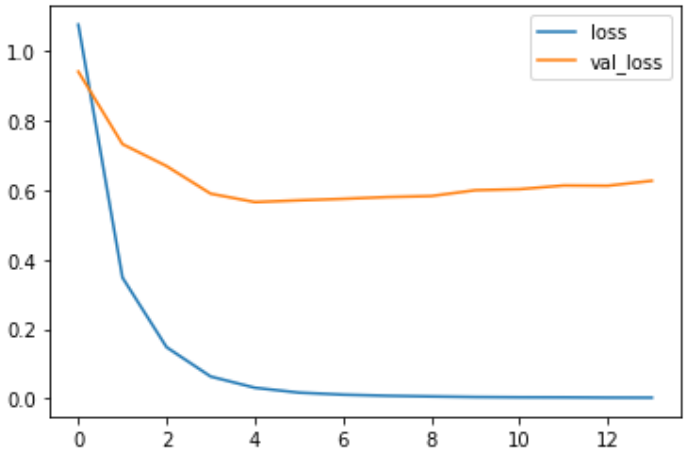


Рисунок 4.11 – График изменения loss с ходом эпох

Графики сохраняют общую тенденцию к быстрому обучению и переходу в стагнацию. Это обусловлено тем, что обучение нейросети перезапускалось с уже откалиброваны весами с других итераций обучения, поэтому модель быстро достигла максимально возможного качества и точности, и спустя некоторое время в стагнации процесс обучения был остановлен после 13 эпохи callback-функцией EarlyStop.

При отобранном наборе данных и списку классов, это был наилучший полученный результат точности модели, поэтому эта версия модели была принята за финальную.

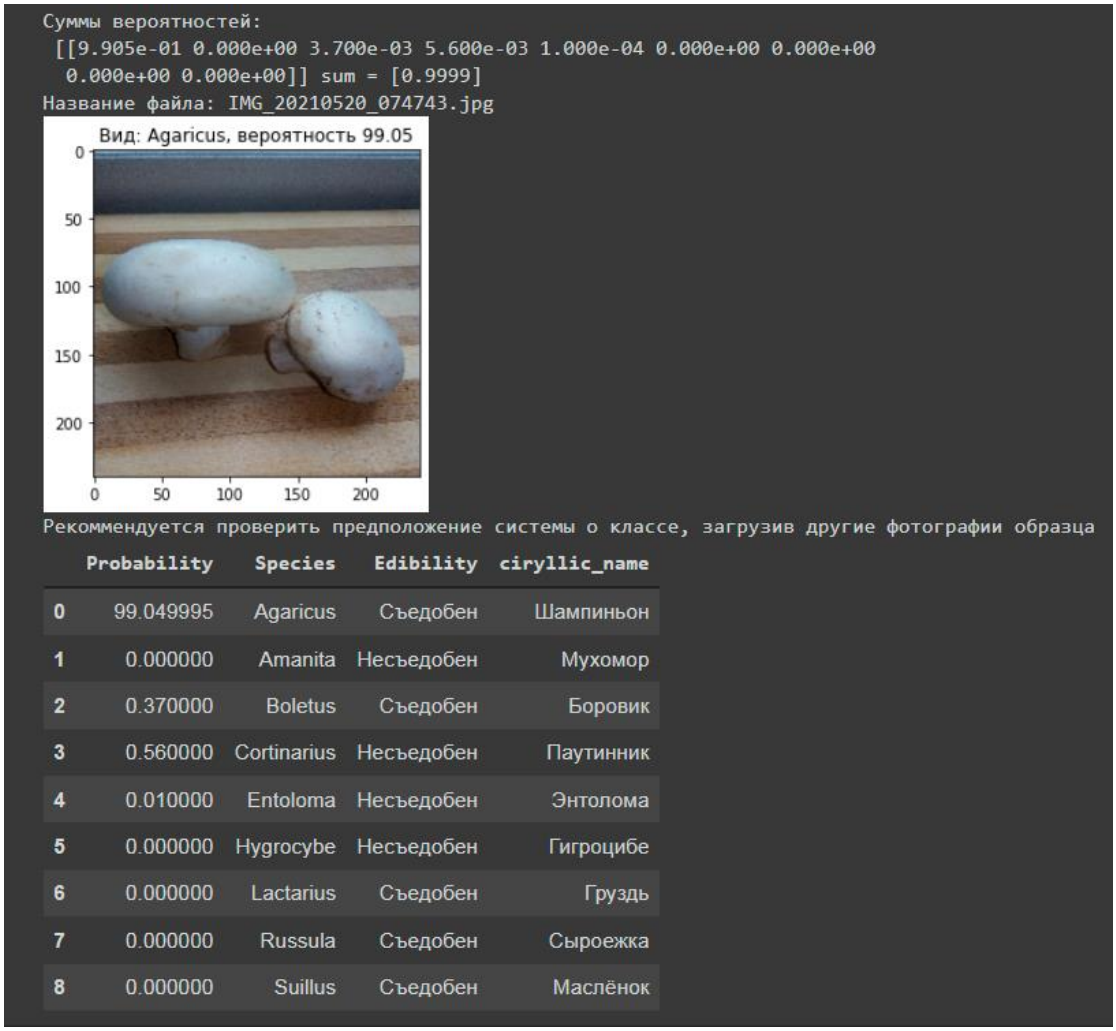
Основным путем для улучшения результатов модели является добавления новых данных, с проверкой влияния на качество обучения различных комбинаций аугментаций.

5 ДЕМОНСТРАЦИЯ РАБОТЫ СИСТЕМЫ

В систему можно загрузить любое изображение, оно будет автоматически преобразовано в нужный формат, после чего с помощью метода predict() оно будет передано в модель, которая использует оптимизированные весовые коэфициенты, чтобы получить вектор взвешенных сумм признаков пропущенных через softmax, и вывести их в виде вероятностей принадлежности разным классам.

Изображению будет присвоен класс с максимальной вероятностью, однако пользователю также будут показаны вероятности других классов, названия на двух языках, и статус съедобности всех видов. Система также предложит загрузить больше изображений классифицируемого образца с других ракурсов, чтобы была возможность распознать другие признаки.

Примеры выводов системы (Рисунок 5.1, Рисунок 5.2):



	Probability	Species	Edibility	ciryllic_name
0	99.049995	Agaricus	Съедобен	Шампиньон
1	0.000000	Amanita	Несъедобен	Мухомор
2	0.370000	Boletus	Съедобен	Боровик
3	0.560000	Cortinarius	Несъедобен	Паутинник
4	0.010000	Entoloma	Несъедобен	Энтолома
5	0.000000	Hygrocybe	Несъедобен	Гигроцибе
6	0.000000	Lactarius	Съедобен	Груздь
7	0.000000	Russula	Съедобен	Сыроежка
8	0.000000	Suillus	Съедобен	Маслёнок

Рисунок 5.1 – Пример классификации пользовательского изображения 1

Суммы вероятностей:
 [[0.2112 0.3966 0.0123 0.0071 0.3267 0.011 0.0306 0.0011 0.0034]] sum = [1.]
Название файла: 1ce8e543-b333-46fe-8c93-2494e4ac109d.jpg

Рекомендуется проверить предположение системы о классе, загрузив другие фотографии образца

	Probability	Species	Edibility	ciryllic_name
0	21.119999	Agaricus	Съедобен	Шампиньон
1	39.660000	Amanita	Несъедобен	Мухомор
2	1.230000	Boletus	Съедобен	Боровик
3	0.710000	Cortinarius	Несъедобен	Паутинник
4	32.670002	Entoloma	Несъедобен	Энтолома
5	1.100000	Hygrocybe	Несъедобен	Гигроцибе
6	3.060000	Lactarius	Съедобен	Груздь
7	0.110000	Russula	Съедобен	Сыроежка
8	0.340000	Suillus	Съедобен	Маслёнок

Рисунок 5.2 – Пример классификации пользовательского изображения 2

6 РАЗВИТИЕ И ПЕРСПЕКТИВЫ

6.1 Система, как MVP продукт

Современное состояние IT-индустрии и IT-рынка предполагает реализацию многих информационных систем, как дружелюбных к пользователю продуктов. Управление продуктом, планирование его жизненного цикла, контроль этапов разработки – важные аспекты в работе над любой ИС.

Представленная в работе экспертная система имеет широкий потенциал к развитию, но и на текущий момент она может называть продуктом, который соответствует концепции MVP (Minimal Viable Product, Минимальной жизнеспособный продукт).

«MVP — тестовая версия товара, услуги или сервиса с минимальным набором функций (иногда даже одной), которая несет ценность для конечного потребителя.»[12]

Возможность только отправить изображение и получить класс на нём (вид гриба), а также вероятность съедобности, - это минимальная функциональность, которую можно назвать продуктом, в случае представления доступа сторонним пользователям.

KPI такого продукта – это число проверяемых пользователями изображений, которое будет говорить о востребованности сервиса.

«MVP создают для тестирования гипотез и проверки жизнеспособности задуманного продукта, насколько он будет ценным и востребованным на рынке. Результаты тестирования минимально жизнеспособного продукта и обратная связь от целевой аудитории помогают понять, стоит ли развивать проект дальше, какие изменения следует внести в стратегию, а что оставить в первоначальном виде.»[12]

Представленная информационная система по распознаванию образов грибов и их классификации в соответствии может быть улучшена в первую очередь посредством ввода дружелюбного к массовому пользователю UI, что может быть реализовано следующими способами:

- перенос системы в пользовательское приложение (Web, desktop, или мобайл) ;
- подключение системы к сторонним площадкам через API в виде бота (например, Telegram или Discord).

Была выдвинута гипотеза о том, будет ли востребован разработанный сервис и будет ли он актуален в виде веб-сайта , для чего был проведен соответствующий онлайн-опрос.

В результате на вопрос:

"Если бы у вас под рукой был сайт для сбора грибов, вы бы чаще собирали грибы?"

из 21 ответов было получено 17 ответов «да», что уже была расценено, как минимальное обоснование в актуальности подобной идеи и следующего предположения по тому, как её реализовывать.

Ссылка на результаты опроса - https://docs.google.com/forms/d/1Fmzlb8waRKRpURX5h5LQpp30E69jKnHx3VTmMmQ0Ef4/edit#responses

Скриншоты результата опроса в приложении В.

6.2 Проектирование веб-приложения

MVP предполагает выдвижение гипотез, на создание которых не требуется большое количество ресурсов, поэтому веб-сайт был разработан в статичном виде, без подключения и взаимодействия с бэк-ендом. Также сайт, представляющий собой front-end-часть системы, разрабатывался на базе конструктора Wix.com, что упростило работу.

В ходе разработки сайта были учтены минимально необходимые свойства продукта для соответствия рынку, такие как:

- название (бренд): FungiAI (произн. *фангайаи*; от англ. *fungi* – грибы; *AI* – ИИ, искусственный интеллект);
- минимально-необходимый для взаимодействия графический интерфейс, отвечающий базовым правилам проектирования UI[12] (т.к. естественность, простота, дружественность и др.);
- базовая информация и описание продукта (системы) ;
- предупреждение о рекомендательном характере системы;
- форма для взаимодействия с back-end часть.

Главная функциональность веб-сайта: предоставить пользователю форму для отправки файла-изображения, после загрузки и отправки которого отображаются данные по классу этого изображения (вид гриба), съедобен он или нет, а также вероятность его съедобности.

Веб-сайт позволит системе стать более доступной для пользователя и проверить, насколько продукт будет востребован среди массового пользователя в таком формате.

Также на веб-странице представлена форма обратной связи, которая даёт возможность узнать о том, как стоить изменить направление реализации и развития проекта.

Ссылка на сайт – **https://khripkovkonstantin.wixsite.com/fungiai**

Скриншоты сайта в приложении Г.

Сама система не была подключена к веб-сайту, т.к. он служит лишь представлением потенциала экспертной системы в виде готового продукта в виде веб-приложения.

Созданный веб-сайт статичен, но может быть полноценно использован в работе и стать front-end частью системы, либо служить дизайн-проектом для интеграции системы на базе других платформ.

В случае добавление сайта в систему потребуется расширить инфраструктуру системы: подключить базу данных и написать веб-сервисы. Для создания такого приложения можно воспользоваться фреймворками Flask (бэк-енд) и React (фронт-енд).

Взаимодействие бэк-енда и фронт-енда будет работать на основе RESTful веб-сервиса, который работает по модели клиент-сервер. REST-архитектура соотносится с HTTP, ключевым понятием в ней являются URI-ресурсы. Клиентские запросы отправляются на URI специальными методами из HTTP (GET, POST, PUT, DELETE) и таким образом взаимодействуют с ними.

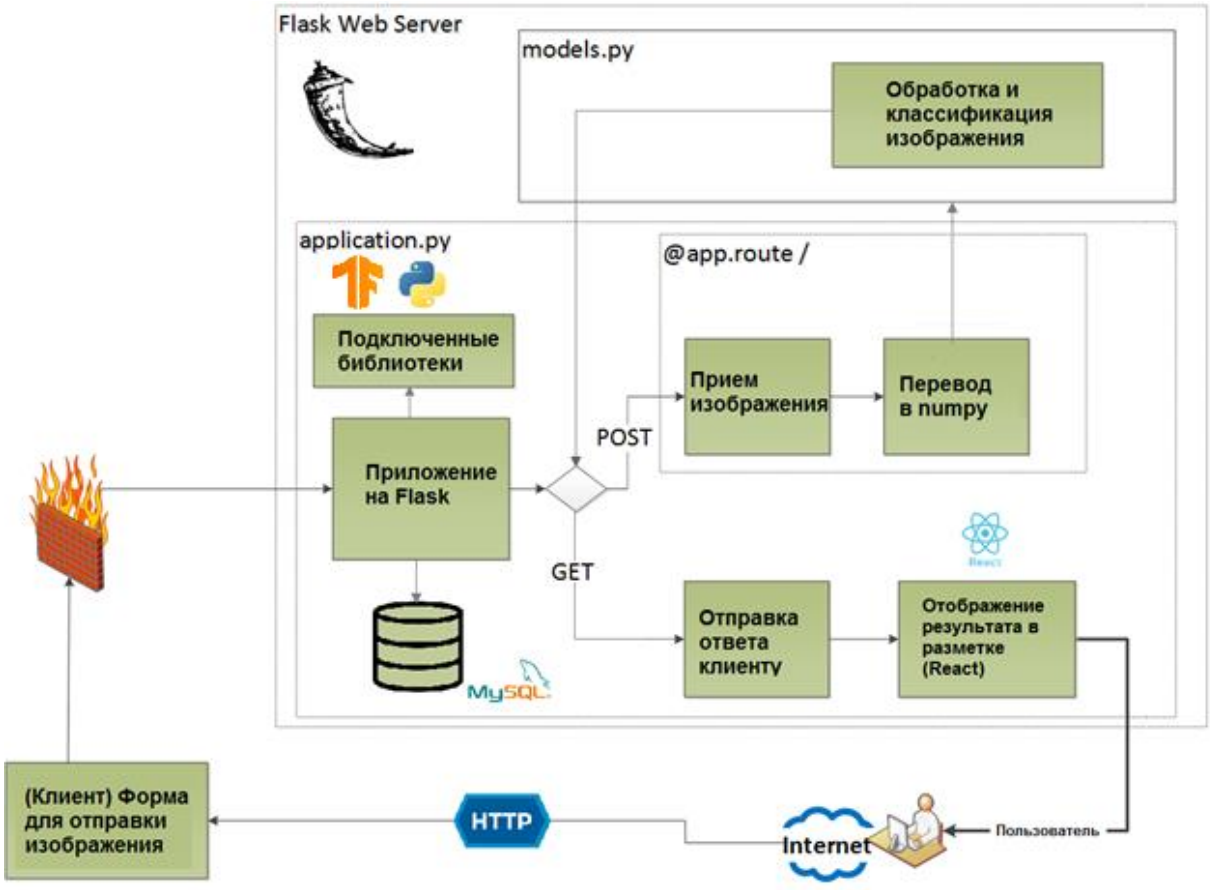


Рисунок 6.1 – Возможная инфраструктура системы в случае будущей разработки веб-приложения

6.3 MVP как процесс

Концепция MVP не предполагает законченный продукт, это скорее процесс, представляющий собой непрерывную генерацию идей, анализ рисков и проведение незатратных эксперементов.

«An MVP is a process that you repeat over and over again: Identify your riskiest assumption, find the smallest possible experiment to test that assumption, and use the results of the experiment to course correct.» «MVP - это процесс, который вы повторяете снова и снова: Вы определяете свое самое рискованное предположение, находите возможный небольшой эксперимент для проверки этого предположения и используете результаты эксперимента для правильного курса (развития).»[13]

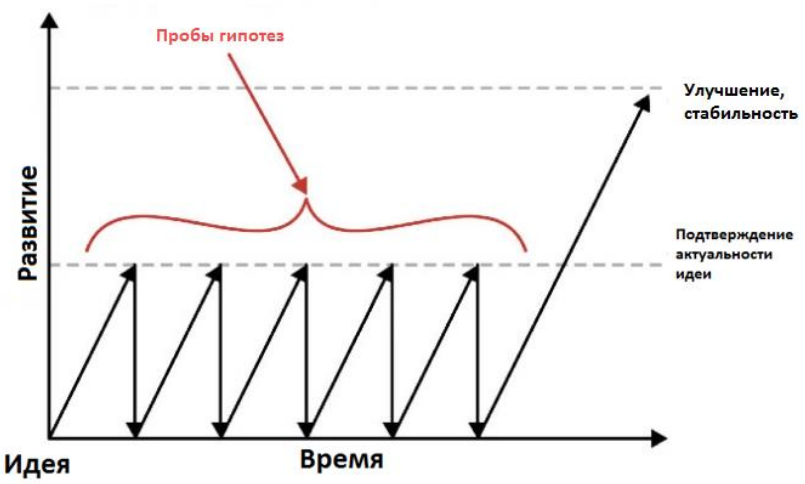


Рисунок 6.2 – Развитие MVP продукта со временем

Эта концепция была применена многими известными и популярными на сегодня IT-компаниями[14], и также используется многими современными стартапами, что видно по возрастающему интересу к MVP.

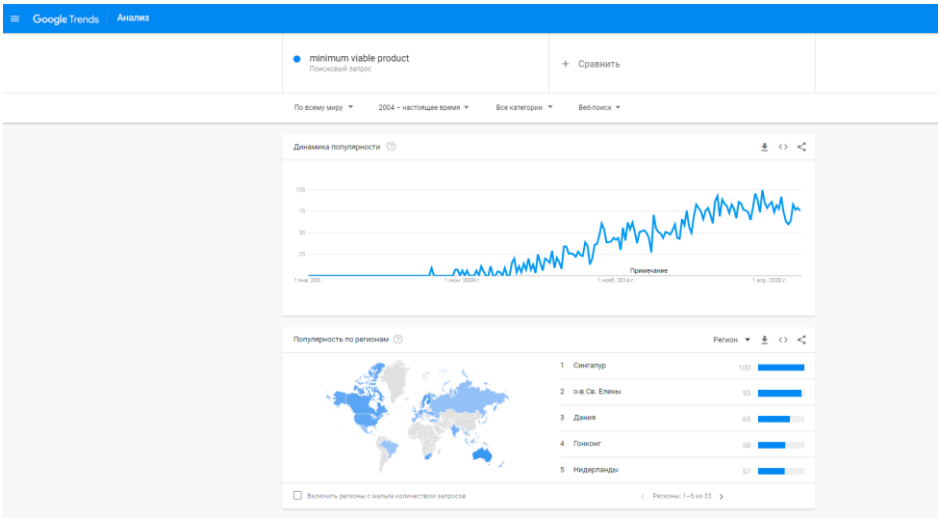


Рисунок 6.3 – Возрастание интереса к MVP со временем

Гипотезы выдвигаются непрерывно и обращать внимание на самые разные аспекты управления продуктом. Они могут быть о функциональном дополнении продукта, о выходе на новые платформы, о монетизации и т.д.

6.4 Пример гипотез по развитию продукта

Гипотеза: Для поддержания удобства доступа к системе, ей необходимы поддерживаемы версии на самых разных платформах, но в первую очередь нужно добавить мобильную версию.

Чтобы проверить актуальность такой идеи, можно создать шаблоны мобильной версии, и провести схожий опрос, как с веб-версии.

Другая гипотеза: есть предположение о том, что система может стать более комплексной и присоединить в себе другую близкую к сбору грибов предметную область – походы и кемпинг. Что может быть добавлено или улучшено?

Одна из идей: добавить карты и GPS навигация в загородной области, что достаточно актуально, потому что хорошие GPS-навигаторы с детализованными картами дорогие, а популярные приложения навигации (Google Maps, 2GIS) не включают подробные внегородские карты отдельных регионов.

Это хорошо сходится с идеей сбора грибов, ибо и то, и то нацелено на любителей выбираться на природу.

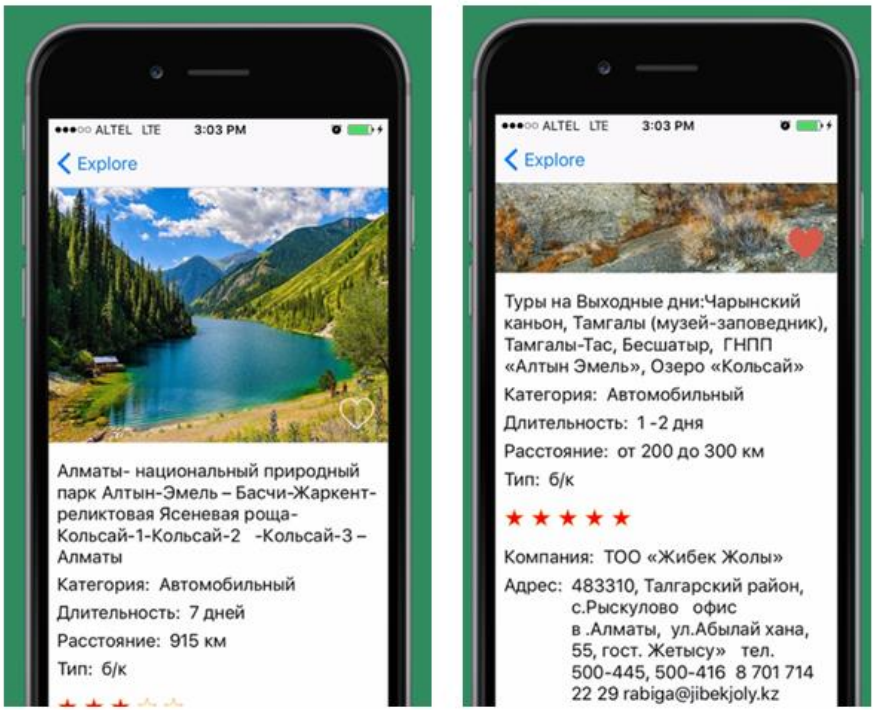


Рисунок 6.4 – Шаблоны возможной мобильной версии

При работе над продуктом также нужно учитывать то, как оно может начать приносить доход. Можно подвести гипотезы о различных способах монетизации, например:

1) Интегрированная реклама: продакт-плейсмент (PP), нативные интеграции;
2) Привлечение инвестиции сторонних компаний, например, реализующих продажу походного инвентаря, или занимающихся локальным туризмом.

Первое предположение можно проверить, внедряя те или иные механизмы и проверяя реакцию пользователей.

Оценить потенциал второй идею можно через проведение презентаций приложения на публичных или закрытых конференциях, по которым можно оценить интерес участников рынка.

В любом случае, даже если какая-либо из названных гипотез не подтвердится, то она не требовали затрат каких-либо ресурсов, что значительно снижает риски и негативные последствия при таких исходах, и продукт может просто начать развиваться в другом направлении.

ЗАКЛЮЧЕНИЕ

Целью работы была разработка экспертной системы, которая помогает пользователю классифицировать вводимые изображения, а именно фотографии грибов.

В рамках работы над проектом была подробно изучена предметная область и спроектирована архитектура модели глубинной свёрточной нейронной сети для распознавания детальных морфологических признаков грибов.

В результате, была достигнута точность в 82% при классификации 9 видов грибов, распространенных в Южном Казахстане. В условиях сложности поставленной задачи определения видов, которые только частично классифицируются по внешнему виду, подобную точность можно считать удовлетворительной.

При внесении в систему своего образца, пользователю будет отправлен ответ, в котором сообщается вид и статус съедобности распознанного образца.

Система располагает уникальным набором данных о грибах Казахстана, аналога которому нет. Этот набор данных может быть использован для улучшения разработанной модели в будущем, а также предоставляет возможность другим разработчикам создавать и обучать свои модели нейронных сетей на его основе.

Также помимо самой системы был проработан план по её потенциальному развитию, как для текущей функции через пополнение набора данных и корректировку архитектуры модели, так и по другим аспектам, например по расширению функциональности и доступности.

Ссылка на репозиторий проекта на github:
https://github.com/forcelumerence/KZ_mushrooms_classification

ПЕРЕЧЕНЬ ТЕРМИНОВ

Машинное обучение – науки об алгоритмах, которые сами настраиваются на основе введенных данных.

Распознавание образов – использование методов машинного обучения, созданных для настройки связей между признаками и явлениями.

KPI (Key Performance Indicator) – это показатель достижения успеха в определенной деятельности или в достижении определенных целей.

Компьютерное зрение – теория и технология создания машин, которые могут производить обнаружение, отслеживание и классификацию объектов.

Таксономия – учение о принципах и практике классификации и систематизации сложноорганизованных иерархически соотносящихся сущностей.

Морфология, наружная морфология (в биологии) – изучает внешнее строение (форму, структуру, цвет, образцы) организма, таксона или его составных частей.

Гимениальная пластинка – поверхность под шляпкой гриба.

Дата-инженерия (англ. data engineering) – смесь аналитики данных и методов data science. Дата-инженеры отвечают за извлечение, преобразование, загрузку данных и их обработку.

Разведочный анализ данных (англ. exploratory data analysis, EDA) — анализ основных свойств данных, нахождение в них общих закономерностей, распределений и аномалий.

Метаданные – информация о другой информации, или данные, относящиеся к дополнительной информации о содержимом или объект

Гиперпараметры – это параметры для управления процессом обучения, значения которых устанавливаются перед запуском этого процесса.

Градиентный спуск – это алгоритм оптимизации, используемый для минимизации некоторой функции путем итеративного движения в направлении самого крутого спуска, определяемого отрицательным значением градиента. В глубоком обучении градиентный спуск используется для обновления параметров модели.

Парсинг – автоматизированный процесс сбора данных с сайтов

Батч – пакет наблюдений определенного размера

TFLOPS, teraflops – мера производительности, 1 терафплос = 1 триллион (10^12) операций с плавающей точкой в секунду

CPU (англ. central processing unit) - центральный процессор

Минимально жизнеспособный продукт (minimum viable product, MVP) — продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Классификация грибов [Электронный ресурс]. - https://www.britannica.com/science/fungus/Outline-of-classification-of-fungi
2. Classification, names & identification [Электронный ресурс]. - https://www.anbg.gov.au/fungi/classification-names-identification.html
3. Outline of classification of fungi [Электронный ресурс]. - https://www.britannica.com/science/fungus/Outline-of-classification-of-fungi
4. ImageNet Large Scale Visual Recognition Challenge 2015 [Электронный ресурс]. - https://image-net.org/challenges/LSVRC/2015/
5. Документация Beautiful Soup [Электронный ресурс]. - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
6. pandas documentation [Электронный ресурс]. - https://pandas.pydata.org/docs/
7. Документация TensorFlow [Электронный ресурс]. - https://www.tensorflow.org/api_docs
8. Документация Keras [Электронный ресурс]. - https://ru-keras.com/
9. A Gentle Introduction to the ImageNet Challenge (ILSVRC) [Электронный ресурс]. - https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/
10. Ложные грибы [Электронный ресурс]. - https://gribnik.info/category/lozhnye-griby
11. Google Trends [Электронный ресурс]. - https://trends.google.ru/trends/
12. ИНТЕРФЕЙСЫ КОМПЬЮТЕРНЫХ СИСТЕМ [Электронный ресурс]. - https://libr.aues.kz/facultet/fit/kt/10/umm/kt_1.htm
13. MVP: что это такое и как работает? [Электронный ресурс]. - https://habr.com/ru/company/productstar/blog/508892/
14. A Minimum Viable Product Is Not a Product, It's a Processhttps://www.ycombinator.com/library/4Q-a-minimum-viable-product-is-not-a-product-it-s-a-process
15. 11 Standout Examples of Minimum Viable Products [Электронный ресурс]. - https://myva360.com/blog/examples-of-minimum-viable-products
16. Wikipedia, The Free Encyclopedia [Электронный ресурс]. - https://trends.google.ru/trends/explore?date=all&q=minimum%20viable%20product
17. Stack Overflow [Электронный ресурс]. - https://stackoverflow.com

ПРИЛОЖЕНИЕ А

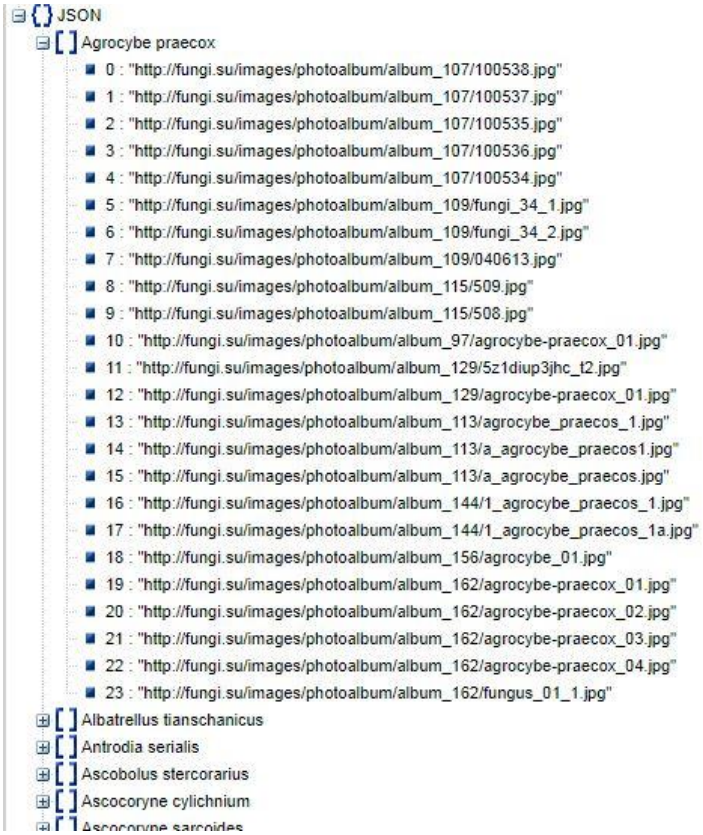


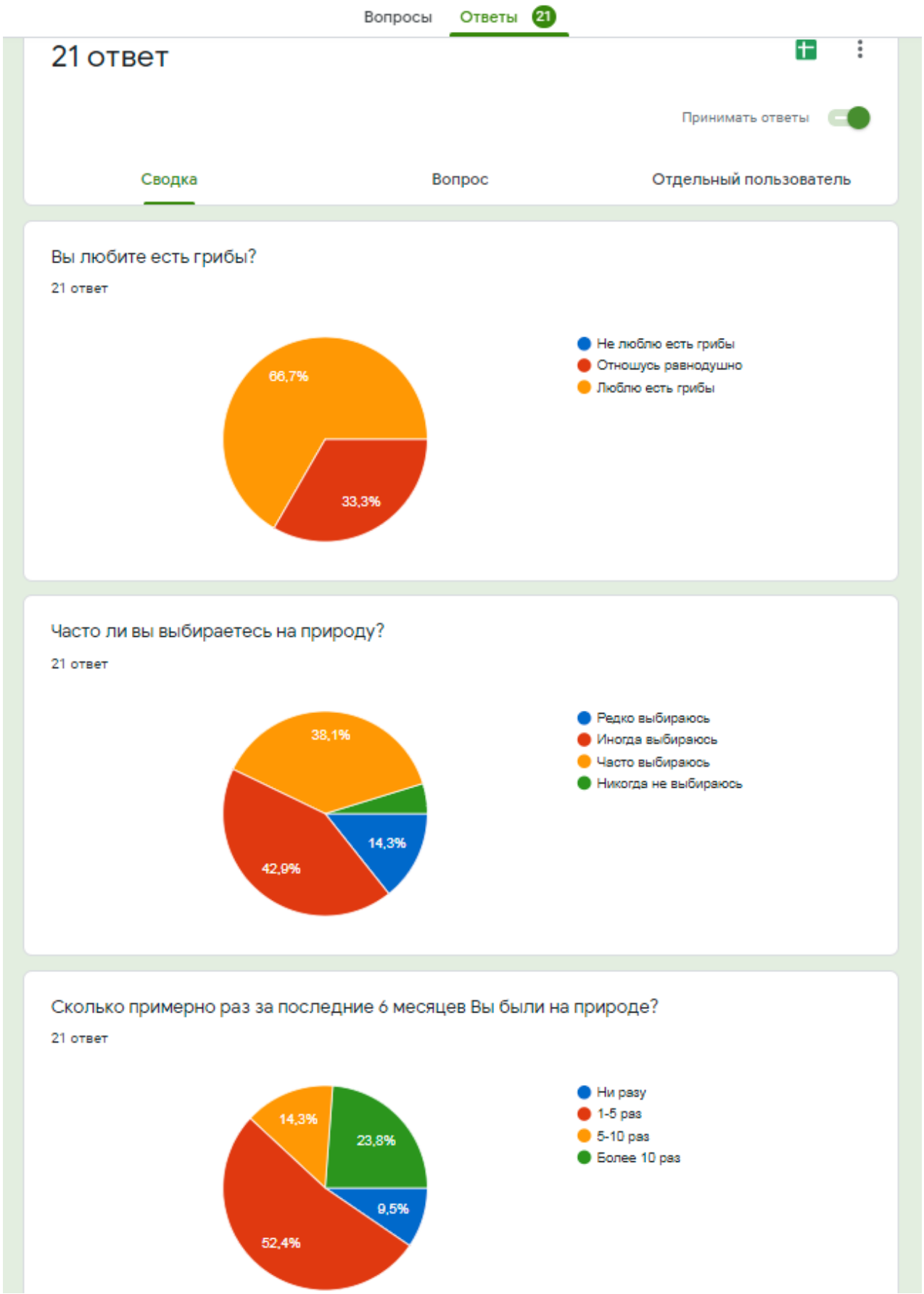
Рисунок – JSON файл с ссылками на файлы изображений

ПРИЛОЖЕНИЕ Б

Рисунок – Примеры удаляемых из датасета образцов

Рисунок – Примеры удаляемых из датасета образцов

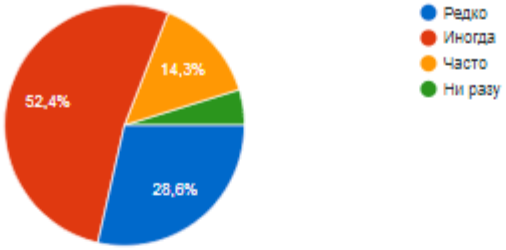
ПРИЛОЖЕНИЕ В



ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ В

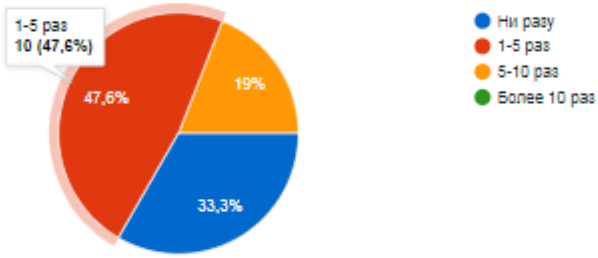
Часто ли вам попадались грибы на природе?

21 ответ



Сколько примерно раз за последние 6 месяцев Вы встречали растущие грибы?

21 ответ

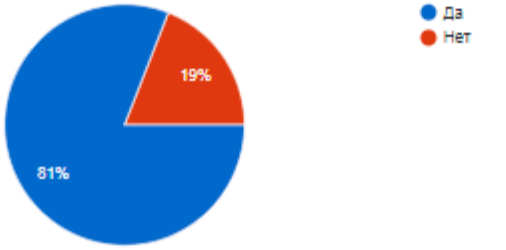


Какое отношение у вас к растущим на природе грибам?

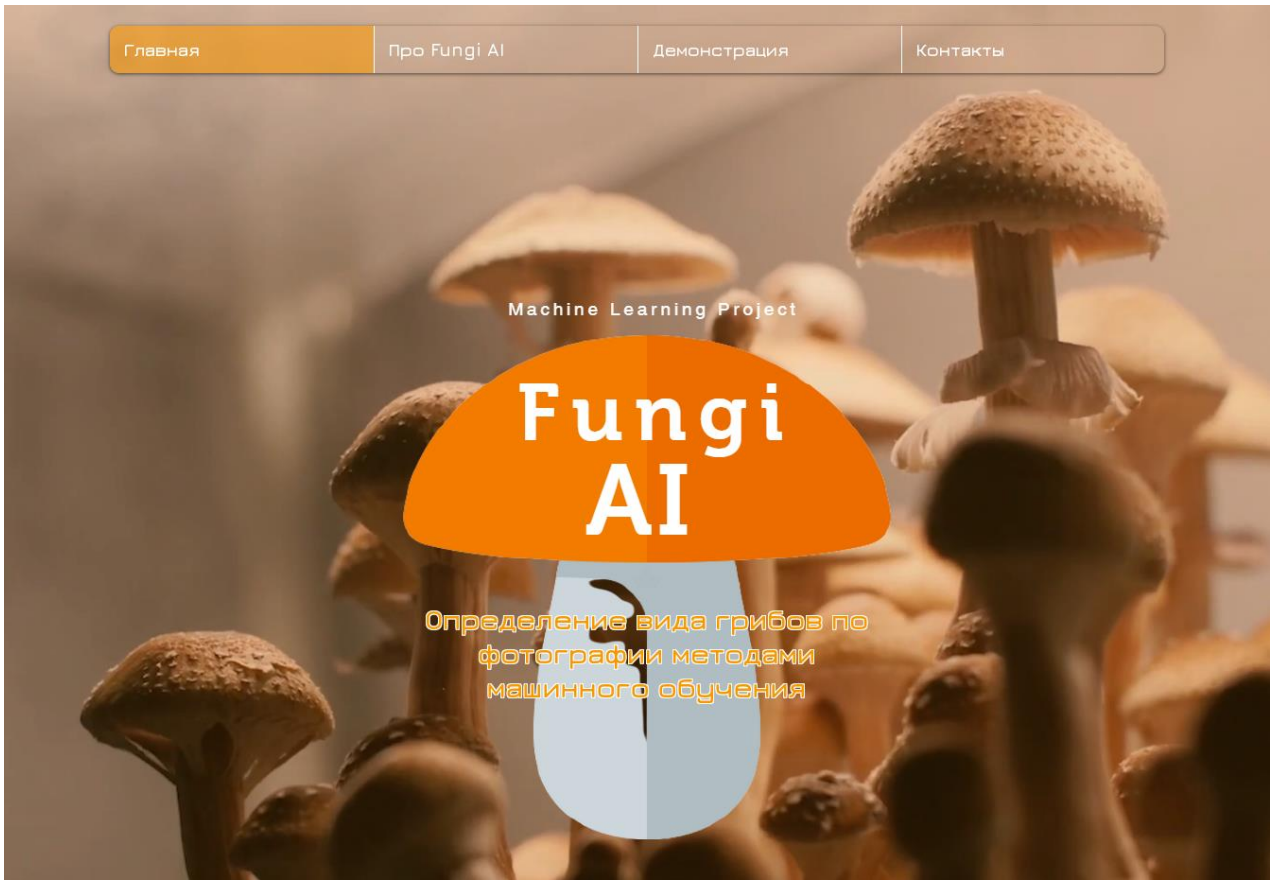
21 ответ

Если бы у вас под рукой был сайт для сбора грибов, вы бы чаще собирали грибы?

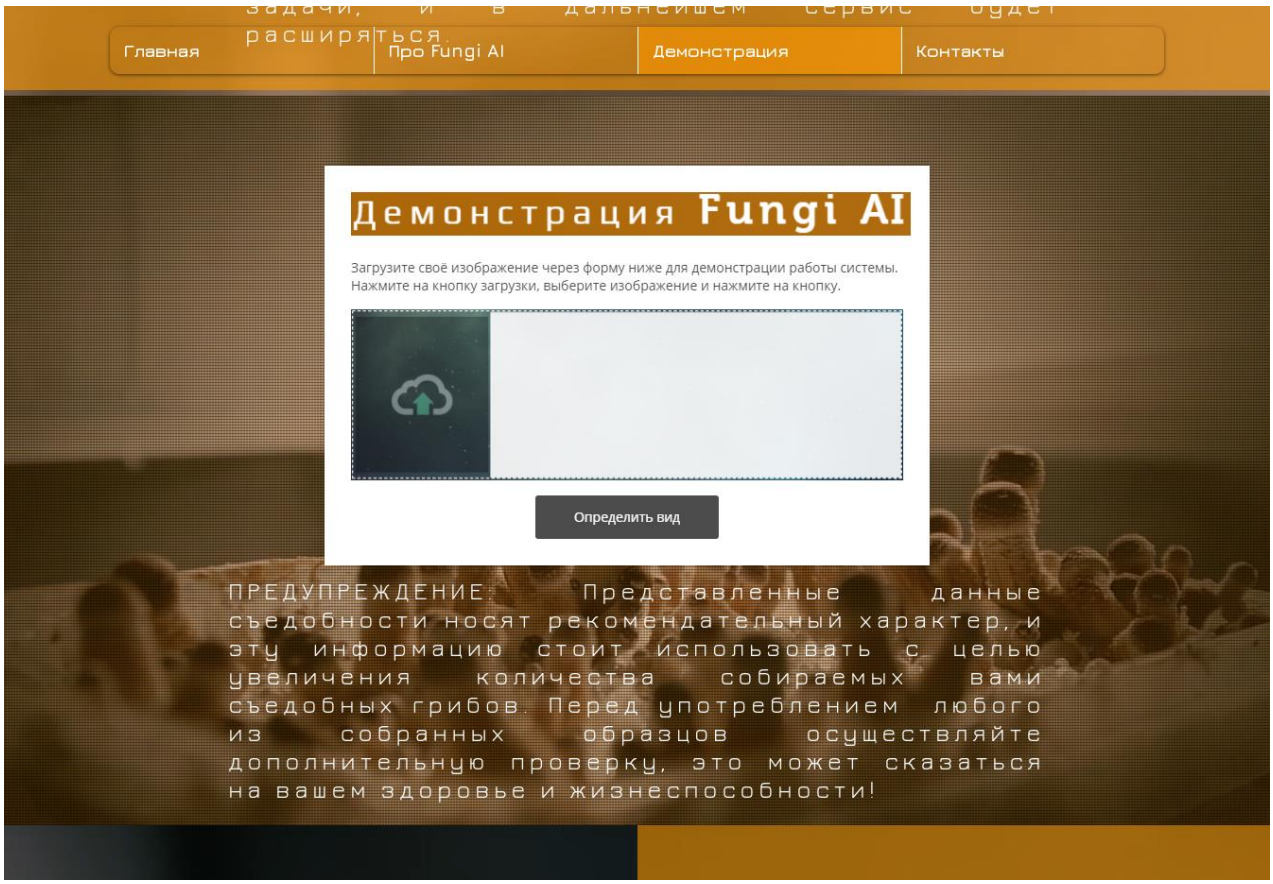
21 ответ

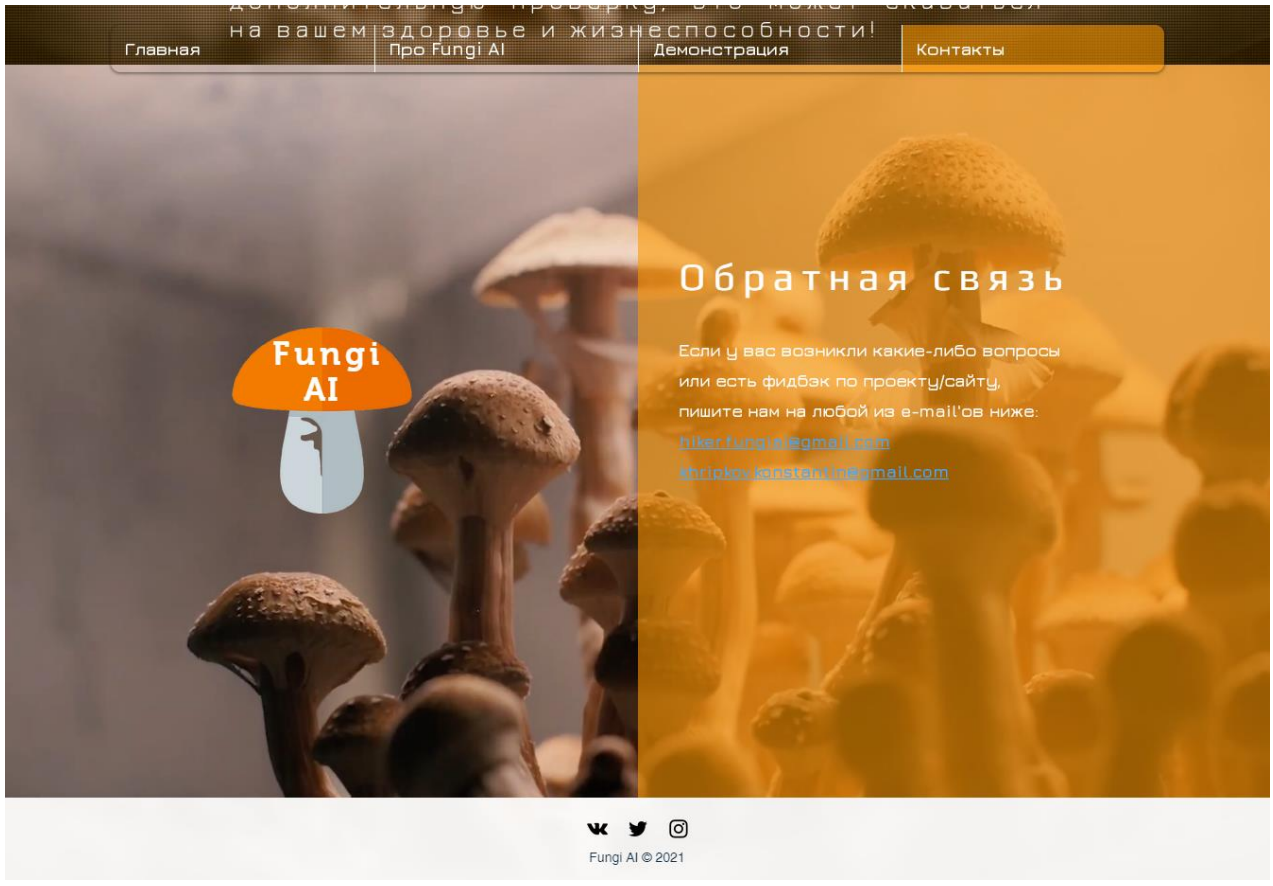


ПРИЛОЖЕНИЕ Г



ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Г


задачи, и в дальнейшем сервис будет расширяться.
Главная
Про Fungi AI
Демонстрация
Контакты
Демонстрация Fungi AI
Загрузите своё изображение через форму ниже для демонстрации работы системы.
Нажмите на кнопку загрузки, выберите изображение и нажмите на кнопку.
Определить вид
ПРЕДУПРЕЖДЕНИЕ: Представленные данные съедобности носят рекомендательный характер, и эту информацию стоит использовать с целью увеличения количества собираемых вами съедобных грибов. Перед употреблением любого из собранных образцов осуществляйте дополнительную проверку, это может сказаться на вашем здоровье и жизнеспособности!


на вашем здоровье и жизнеспособности!
Главная
Про Fungi AI
Демонстрация
Контакты
Fungi AI
Обратная связь
Если у вас возникли какие-либо вопросы или есть фидбэк по проекту/сайту, пишите нам на любой из e-mail'ов ниже:
hiker.fungi@gmail.com
khripko.konstantin@gmail.com
Fungi AI © 2021

ПРИЛОЖЕНИЕ Д

```
# -*- coding: utf-8 -*-
"""mushrooms_project.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1DW0juvxjkDSSnjBtf5J9QuTH9sq-JsUv

### Подключение к Google Диску, архивация и загрузка файлов на локальную машину
"""

from google.colab import drive
drive.mount('/content/drive')

from google.colab import files

# архивация собранных изображений в zip-архив
#!zip -r /content/kz_mushrooms_data.zip /content/Mushrooms_scrapped_images

# скачивание на локальный диск

#files.download("/content/kz_mushrooms_info_full.csv")
#files.download("/content/kz_mushrooms_info.csv")
#files.download("/content/images_links_data.json")
#files.download("/content/kz_mushrooms_data.zip")

"""# 1. Сбор данных

### Подключение пакетов
"""

import csv
import os

import traceback                    #вывод трассировки при вызове исключений

import requests                     #http-запросы
from bs4 import BeautifulSoup       #парсинг html-страниц

import pandas as pd                 #обработка данных
import numpy as np                  #numpy массивы

"""## Парсинг метаданных с веб-страниц списка видов грибо, распространенных в Казахстане
http://fungi.su/infusions/advanced_articles_sort/mycota_reg.php
"""

page_number = 1

for rowstart in range (0, 480, 30):
  url = 'http://fungi.su/infusions/advanced_articles_sort/mycota_reg.php?&rowstart=%d'
% (rowstart) # url страниц списка видов
  page = requests.get(url)
  filename = 'page%s.html' % (page_number)
  page_number += 1
  with open(filename, 'w') as output_file:
    output_file.write(page.text)
```

Алматы 2021

```
urls = []
titles_rus = []
titles_eng = []
edibility = []
divisio = []
classis = []
ordo = []
familia = []
genus = []
location = []
data_labels = [edibility, divisio, classis, ordo, familia, genus]

def parse_data(data, tag):
  data.append(tag.text.strip())

for rowstart in range (0, 480, 30):
  url = 'http://fungi.su/infusions/advanced_articles_sort/mycota_reg.php?&rowstart=%d'
% (rowstart) # url страниц списка видов
  page = requests.get(url) #запрос страницы по заданному URL

  soup = BeautifulSoup(page.text) #прогон страницы для получения всего находящейся на
ней HTML-разметки
  tds = soup.find_all('td', {'class': 'tbl1_a'}) #поиск всех тегов <td>, в каждом из
которых находится информация по одному виду грибов

  for i in range(0, len(tds)):
      atags = tds[i].find_all('a', class_='side') #теги внутри td никак уникально не
классифицированы, поэтому делается поиск по всем тегам <a> класса side

      #извлечение url с ссылкой на страницу вида, который находится в первом теге <a>
      href = atags[0].get('href')
      urls.append(href)
      print(href)

      #информация о названиях находится в первух двух тегах <strong> (жирный шрифт)
      strongtags = tds[i].find_all('strong')
      titles_rus.append(strongtags[0].text.strip())
      print(strongtags[0].text.strip())
      titles_eng.append(strongtags[1].text.strip())
      print(strongtags[1].text.strip())

      #сбор данных по видам грибов, распространенных в Казахстане, они находятся в
тегах <a> с 2 по 6
      for c in range(1, 7):
        parse_data(data_labels[c-1], atags[c])

      #данные о расположении находится в тегах <a> начиная с 7-го, они объединяются в
одну строку
      loc = ""
      for k in range(7, len(atags)):
        loc+=str(atags[k].text.strip()) + " "
      location.append(loc)

      for j in range(1, len(atags)):
         print(atags[j].text.strip())
      print('\n')

#извлечение из ссылок ID статей
articles_ids = []
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
for i in range(0, len(urls)):
  articles_ids.append(int(urls[i].replace('../../articles.php?article_id=', '')))

"""
### Внесение и сохранение данных в формате датафреймов"""

#внесение полученных данных в пустой датафрейм
kz_mushrooms_info_full = pd.DataFrame()

kz_mushrooms_info_full["title_rus"] = np.array(titles_rus)
kz_mushrooms_info_full["title_eng"] = np.array(titles_eng)
kz_mushrooms_info_full["edibility"] = np.array(edibility)
kz_mushrooms_info_full["divisio"] = np.array(divisio)
kz_mushrooms_info_full["classis"] = np.array(classis)
kz_mushrooms_info_full["ordo"] = np.array(ordo)
kz_mushrooms_info_full["familia"] = np.array(familia)
kz_mushrooms_info_full["genus"] = np.array(genus)
kz_mushrooms_info_full["location"] = np.array(location)
kz_mushrooms_info_full["links"] = np.array(urls)
kz_mushrooms_info_full['article_id'] = np.array(articles_ids)

kz_mushrooms_info_full

#внесение полученных данных в пустой датафрейм
kz_mushrooms_info = pd.DataFrame()

#для системы нужны только следующие данные: название на англ. и русском языках,
съедобность, род гриба, и ссылка на страницу
kz_mushrooms_info["title_rus"] = np.array(titles_rus)
kz_mushrooms_info["title_eng"] = np.array(titles_eng)
kz_mushrooms_info["edibility"] = np.array(edibility)
kz_mushrooms_info["genus"] = np.array(genus)

kz_mushrooms_info

kz_mushrooms_info_full.to_csv('kz_mushrooms_info_full.csv') #запись датафреймов в
формате csv
kz_mushrooms_info.to_csv('kz_mushrooms_info.csv')

"""---
## *Сбор датасета изображений*

### Парсинг ссылок на файлы изображений со страниц отдельных видов
http://fungi.su/articles.php?article_id=2162
"""

img_links_dict = {}

for title in titles_rus:
  #в адресе на фрейм с ссылками на изображения в качестве идентификатора используются
названия видов на кириллице
  url = 'http://fungi.su/infusions/advanced_articles_sort/phfa.php?stext=%s'% (title)
#URL страницы вида

  page = requests.get(url)         #запрос страницы по заданному URL
  soup = BeautifulSoup(page.text, 'html.parser') #прогон страницы для получения всего
находящейся на ней HTML-разметки
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
 #print(soup.prettify())            #вывод полученного HTML

  img_tags = soup.find_all('a')                        #все изображения на странице
находятся в тегах <a>, у каждого есть атрибут href с ссылкой на галерею

  if len(img_tags) > 30:                               #на странице вида показывается
не более 30 изображений
    url = img_tags[30].get('href')                     #чтобы увидеть все сайт
предлагает перейти на отдельную страницу по кнопке "Посмотреть все фото",
    page = requests.get(url)                           #к которой прикреплена последня
ссылка на странице (31-ая)
    soup = BeautifulSoup(page.text, 'html.parser')
    img_tags = soup.find_all('a', target='_blank')     #все изображения на странице
находятся в тегах <a> с атрибутом target='_blank', у каждого есть атрибут href с
ссылкой на галерею

  img_links = []

  for i in range(0, len(img_tags)):
    try:
      href = img_tags[i].get('href')                         #ссылки извлекаются из
атрибута href
    except AttributeError as e:                              #если атрибут будет пуст
      print('Атрибут src пуст, тег пропускается, i=', i'; вид:', title)
      traceback.print_exc()
      continue
    except Exception as exception:
      traceback.print_exc()

    #print('Ссылка в галлерею: ', href)
    img_page = requests.get(href)
    soup = BeautifulSoup(img_page.text, 'html.parser')

    img_direct_tag = soup.find('img', title="Нажмите для просмотра оригинального
размера") #поиск в HTML-тексте тега с ссылкой на оригинальное изображение

    try:
      src = img_direct_tag.get('src')                        #в атрибуте src (source)
ссылка на источник изображения на сервере
    except AttributeError as e:                              #если атрибут будет пуст
      print('Атрибут src пуст, тег пропускается, i=', i, '; вид:', title)
      traceback.print_exc()
      continue
    except Exception as exception:
      traceback.print_exc()

    direct_link = 'http://fungi.su/' + src                 #формирование прямой ссылки
    img_links.append(direct_link)
    #print('Прямая ссылка: ', direct_link)

  id = titles_rus.index(title)                  #индекс для ссылки на английское название
  title_eng = titles_eng[id]                                #английское название
  img_links_dict.update({title_eng : img_links})      #словарь, где название вида -
ключ, а ссылки на изображения вида - значения

  print('#', id, ' Вид:', title_eng, '- Число ссылок:', len(img_links))
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
"""### Сохранение полученного словаря с собранными ссылками на файлы изображений в
```

```
формате json """

import json

# Сохранение словаря со структурой {вид : [список ссылок]} в json формате
saved_links_data = img_links_dict.copy()
save_file = open("images_links_data.json", "w")
json.dump(saved_links_data, save_file)
save_file.close()

# Чтение данных из дамп файла

save_file = open("images_links_data.json", "r")
links_dict = save_file.read()
print(links_dict)
save_file.close()

"""### Загрузка  в хранилище изображений из ссылок по соответствующим директориям"""

# создание директории для каждого вида
def create_dir(name):
    dirName = '/content/Mushrooms_scrapped_images/' + name
    os.mkdir(dirName)
    return dirName

# имя загружаемого файла
def filename(url):
    filename = url.split('/')[-1]
    return filename

# имя запрос на получение файла изображения
def get_image(url):
    img_source = requests.get(url)
    return img_source

# загрузка пришедшего изображения с заданными именем в обозначенную директорию
def download_image(filename, image, directory):
    f_name_dir = os.path.join(directory, filename)
    with open(f_name_dir, 'wb') as output_file:
      output_file.write(image.content)

# основная директория

#dirName = 'Mushrooms_scrapped_images'
#os.mkdir(dirName)

i=0

# ключ словаря будет названием директории, в которую буду скачиваться файлы
изображений из ссылок, расположенных в соответсвующем данному ключу списке
for key in img_links_dict:
  i+=1
  print ('#', i, 'Вид: ', key, ', число ссылок: ', len(links_data[key]))
  img_dir = create_dir(key)
  for direct_link in img_links_dict.get(key):
      download_images(filename(direct_link), get_image(direct_link), img_dir)
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
"""### Проверка совпадания именований классов и их количества используемых данных и их
количества по метаданным """
```

```
kz_mushrooms_info_full = pd.read_csv('D:\Downloads\DATA\kz_mushrooms_info_full.csv')
kz_mushrooms_info_full.drop(columns=['links', 'article_id', 'Unnamed: 0'], axis=1,
inplace=True)
kz_mushrooms_info_full

fungi_data = kz_mushrooms_info_full['title_eng']
kz_fungi_names = fungi_data.values.tolist()

data_path = 'D:\Downloads\DATA\KZ_fungi_dataset\kz_fungi_dataset'
fungi_data_classes = os.listdir(data_path)

print(len(fungi_data_classes))

print(len(kz_fungi_names))

list(set(fungi_data_classes) - set(kz_fungi_names))

list(set(kz_fungi_names) - set(fungi_data_classes))

"""# 2. Первичный анализ данных (EDA)"""

kz_mushrooms_info_full = pd.read_csv('/content/kz_mushrooms_info_full.csv')

kz_mushrooms_info_full.drop(columns=['links', 'article_id', 'Unnamed: 0'], axis=1,
inplace=True)

kz_mushrooms_info_full.info()

kz_mushrooms_info_full.isna().sum()

kz_mushrooms_info_full['location'].fillna('ЮК ЦК ВК СК ЗК', inplace=True)

"""Пропущенное значение edibility будет заполнено далее
"""

# уникальные значения каждого столбца
kz_mushrooms_info_full.nunique()


#Грибная систематика
#title - наименование вида (в данном датасете)
#devisio - отдел
#classis - класс
#ordo - порядок
#familia - семейство
#genus - род

"""С каждом ступенью от отдела до вида происходит разделение на в среднем ~2.5-3
подчасти, т.е. 32 отрядя включают 84 семейства (≈2.5 на отряд), 84 семейства 199 родов
(также ≈2.5 на семейств), а 199 родов - 455 видов (≈2.3 на род)."""

# число видов, распространенных в разных регионах Казахстана.
kz_mushrooms_info_full['location'].value_counts()

south_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ЮК')].reset_in
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
dex(drop=True)
print ('Видов в ЮК:', south_kz_subset.shape[0])
north_kz_subset =
```

```
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('СК')].reset_in
dex(drop=True)
print ('Видов в СК:', north_kz_subset.shape[0])
east_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ВК')].reset_in
dex(drop=True)
print ('Видов в ВК:', east_kz_subset.shape[0])
west_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ЗК')].reset_in
dex(drop=True)
print ('Видов в ЗК:', west_kz_subset.shape[0])
central_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ЦК')].reset_in
dex(drop=True)
print ('Видов в ЦК:', central_kz_subset.shape[0])

print(kz_mushrooms_info_full['edibility'].value_counts(), '\n')
print(kz_mushrooms_info_full['edibility'].value_counts(normalize=True), '\n')

kz_mushrooms_info_full['edibility'].value_counts().plot.pie();

"""Съедобен => гриб можно собирать
Условносъедобен => гриб нужно употреблять в приготовленном виде => гриб можно собирать
Несъедобен, ядовит, смертельноядовит, галлюциногенный => гриб не стоит собирать
Распределение видов, рекомендуемых к сбору и наоборот - ≈52% на ≈48% соответственно.
#### Подведение статуса съедобности в бинарный вид
"""

kz_mushrooms_info_full_orig = kz_mushrooms_info_full.copy()

"""Задачей системы является дать рекомендацию к тому собирать найденный гриб или нет,
соответственно все статусы съедобности у видов утилизируются до "съедобен/несъедобен",
которые трактуются как "собирать/не собирать" """

kz_mushrooms_info_full_1 = kz_mushrooms_info_full.copy()
kz_mushrooms_info_full_1 =
kz_mushrooms_info_full_1.loc[kz_mushrooms_info_full_1['edibility'] ==
'Условносъедобен']
kz_mushrooms_info_full_1['edibility'] = 'Съедобен'
kz_mushrooms_info_full.loc[kz_mushrooms_info_full_1.index] = kz_mushrooms_info_full_1

kz_mushrooms_info_full_2 = kz_mushrooms_info_full.copy()
kz_mushrooms_info_full_2 =
kz_mushrooms_info_full_2.loc[(kz_mushrooms_info_full_2['edibility'] == 'Ядовит')
                                        | (kz_mushrooms_info_full_2['edibility'] ==
'Смертельноядовит')
                                        | (kz_mushrooms_info_full_2['edibility'] ==
'Галлюциногенный')]
kz_mushrooms_info_full_2['edibility'] = 'Несъедобен'
kz_mushrooms_info_full.loc[kz_mushrooms_info_full_2.index] = kz_mushrooms_info_full_2

"""Один из видов без статуса съедобности. Проверка по названию:
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
"Съедобность: по постановлению Минздрава России, наряду со Свинушкой тонкой,
причислена к ядовитым грибам. Тем не менее употребляется в пищу после предварительного
отваривания."
```

```
http://fungi.su/articles.php?article_id=670

Из-за ядовитого статуса зачислен в несъедобные, несмотря на свою условную съедобность
"""

kz_mushrooms_info_full_3 = kz_mushrooms_info_full.copy()
kz_mushrooms_info_full_3 =
kz_mushrooms_info_full_3.loc[(kz_mushrooms_info_full_3['edibility'] != 'Съедобен')
                                        & (kz_mushrooms_info_full_3['edibility'] !=
'Несъедобен')]
kz_mushrooms_info_full_3

kz_mushrooms_info_full_3['edibility'] = 'Несъедобен'
kz_mushrooms_info_full.loc[kz_mushrooms_info_full_3.index] = kz_mushrooms_info_full_3

print(kz_mushrooms_info_full['edibility'].value_counts(normalize=True), '\n')
print(kz_mushrooms_info_full['edibility'].value_counts())

kz_mushrooms_info_full['edibility'].value_counts().plot.pie();

"""В результате, среди распространенных в Казахстане видов доля несъедобных грибов
слегка выше

#### Исследование данных для проведения обобщения классов

455 уникальных видов грибов, распространенных в Казахстане, - это большое число
классов для распознавания. С используемым датасетом нейронная сеть никаким образом не
сможет найти и обобщить уникальные паттерны для каждого из видов.
При доступном количестве наблюдений обучить качественную модель невозможно, поэтому
некоторые виды необходимо обобщить.
Также классификация будет происходить только по тем видам, что встречаются в отдельных
регионах, в частности - в Южном Казахстане.
"""

south_kz_subset

"""356 видов представлены в южном Казахстане"""

south_kz_subset.nunique()

print(south_kz_subset['edibility'].value_counts())

"""Отделы, классы и порядки имеют слишком большие отличия признаков и не поддаются
обобщению.
Разница съедобных/несъедобных видов в рамках одного отряда, семейства или рода
показывает, какая из ступеней иерархии в случае обобщения будет наименее критична при
предсказывании её классов и определении съедобности.
"""

south_kz_subset['ordo'].value_counts().head(10)

south_kz_subset[['ordo', 'familia', 'genus', 'edibility']].value_counts()

south_kz_subset['edibility'].groupby(south_kz_subset['ordo']).value_counts(normalize=T
rue)
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
ordo_ratio =
south_kz_subset['edibility'].groupby(south_kz_subset['ordo']).value_counts()
ordo_ratio.unstack().plot.bar(stacked=True, figsize=(10,8), colormap="brg")
```

```
"""Есть 15 отрядов (из 32), в которых есть как съедобные, так и несъедобные виды
грибов.

8-9 (25-28%) отрядов имеют примерно равное соотношение съедобных/несъедобных.

Высокая доля подобных классов сильно повредит качеству выдаваемых предсказаний,т.к.
даже предсказав верный отряд системы не сможет точно сообщить, стоит ли собирать
найденный гриб данного отряда или нет.
"""

south_kz_subset['familia'].value_counts().head(10)

south_kz_subset[['familia', 'genus', 'edibility']].value_counts()

south_kz_subset['edibility'].groupby(south_kz_subset['familia']).value_counts(normaliz
e=True)

familia_ratio =
south_kz_subset['edibility'].groupby(south_kz_subset['familia']).value_counts()
familia_ratio.unstack().plot.bar(stacked=True, figsize=(18,10), colormap="brg")

"""Есть 25 семейств (из 78), в которых есть как съедобные, так и несъедобные виды
грибов.

14-15 (19%) семейств имеют примерно равное соотношение съедобных/несъедобных.

Качество обобщения семейств немного выше, чем у отрядов, но число классов (78) уже
значительно выше.
"""

south_kz_subset['genus'].value_counts()

south_kz_subset[['genus', 'edibility']].value_counts()

south_kz_subset['genus'].groupby(south_kz_subset['familia']).value_counts(normalize=Tr
ue)

genus_ratio =
south_kz_subset['edibility'].groupby(south_kz_subset['genus']).value_counts()
genus_ratio.unstack().plot.bar(stacked=True, figsize=(18,10), colormap="brg")

"""Есть 24 род (из 173), в которых есть как съедобные, так и несъедобные виды грибов.

16-17 (~10%) отрядов имеют примерно равное соотношение съедобных/несъедобных.

## Итого:

*   У родов наименьший доля (10%) неоднозначных с точки зрения неопределенности о
статусе съедобности отделнього случайного найденого вида
*   Присутствует отрицательная корелляция с количество разделений видов на одну
единицу таксономии (больше видов на единице -> меньше доля неоднозначных единиц)
*   Относительно отрядов и семейств происходит увелечение числа классов в 5.25 раз
(173 родов : 32 отряда) и 2.15 раз (173 родов : 78 семейств) соотвественно
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
При доступном количестве данных классификация по родам не может быть проведена без
дополнительного обобщения до семейств.
Следовательно, применяется обобщение видов до отрядов и семейств, с учетом данных о
соотношении съедобности/несъедобности включаемых видов.
```

```
Пример:
1. Если в семействе все или почти все виды съедобны/несъедобны, то они объеденяются в
один класс и маркируются соответствующим статусом съедобности.
2. Если в семействе есть неопределенность, т.е. разница в количестве съедобных и
несъедобных видов не имеет значительного перевеса в одну из сторон, то обобщение
классов будет до входящих в это семейство родов.
3. Обобщение видов до родов по принципу из 1 пункта.

Обобщение проводится вручную среди грибов Южного Казахстана с принятием во внимание
схожести их морфологических признаков по изображениям.
## Подготовка региональных датасетов и датасетов с обобщением до родов/семейств
#### Информация о датасетах
"""

south_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ЮК')].reset_in
dex(drop=True)
print ('Видов в ЮК:', south_kz_subset.shape[0])
north_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('СК')].reset_in
dex(drop=True)
print ('Видов в СК:', north_kz_subset.shape[0])
east_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ВК')].reset_in
dex(drop=True)
print ('Видов в ВК:', east_kz_subset.shape[0])
west_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ЗК')].reset_in
dex(drop=True)
print ('Видов в ЗК:', west_kz_subset.shape[0])
central_kz_subset =
kz_mushrooms_info_full[kz_mushrooms_info_full['location'].str.contains('ЦК')].reset_in
dex(drop=True)
print ('Видов в ЦК:', central_kz_subset.shape[0])

central_kz_subset.groupby(['familia', 'genus',
'edibility']).size().unstack(fill_value=0)

west_kz_subset.groupby(['familia', 'genus', 'edibility']).size().unstack(fill_value=0)

df = east_kz_subset.groupby(['familia', 'genus',
'edibility']).size().unstack(fill_value=0)
print(df.to_string()) #чтобы вывести целиком

df = north_kz_subset.groupby(['familia', 'genus',
'edibility']).size().unstack(fill_value=0)
print(df.to_string())

df = south_kz_subset.groupby(['familia', 'genus',
'edibility']).size().unstack(fill_value=0)
print(df.to_string())

"""#### Разбиение данных по регионам"""
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
south_kz_fungi_names = south_kz_subset['title_eng'].values.tolist()
north_kz_fungi_names = north_kz_subset['title_eng'].values.tolist()
west_kz_fungi_names = west_kz_subset['title_eng'].values.tolist()
east_kz_fungi_names = east_kz_subset['title_eng'].values.tolist()
central_kz_fungi_names = central_kz_subset['title_eng'].values.tolist()
```

```
import shutil
import os
from distutils.dir_util import copy_tree #методы для копирования и перемещения
директорий и файлов

main_directory = "D:\Downloads\DATA\KZ_fungi_dataset\kz_fungi_dataset"
folders_list = os.listdir(main_directory)

region_directory =
"D:\Downloads\DATA\KZ_fungi_dataset\Regional_datasets\south_kz_fungi_dataset"
species_list = south_kz_fungi_names

#будут копироваться только те директории, названия которых есть в списке наименований
видов в регионе
for folder in folders_list:
    if folder in species_list:
        current_directory = os.path.join(main_directory, folder)
        destination = os.path.join(region_directory, folder)
        copy_tree(current_directory, destination)

region_directory =
"D:\Downloads\DATA\KZ_fungi_dataset\Regional_datasets\east_kz_fungi_dataset"
species_list = east_kz_fungi_names

for folder in folders_list:
    if folder in species_list:
        current_directory = os.path.join(main_directory, folder)
        destination = os.path.join(region_directory, folder)
        copy_tree(current_directory, destination)

region_directory =
"D:\Downloads\DATA\KZ_fungi_dataset\Regional_datasets\west_kz_fungi_dataset"
species_list = west_kz_fungi_names

for folder in folders_list:
    if folder in species_list:
        current_directory = os.path.join(main_directory, folder)
        destination = os.path.join(region_directory, folder)
        copy_tree(current_directory, destination)

region_directory =
"D:/Downloads/DATA/KZ_fungi_dataset/Regional_datasets/north_kz_fungi_dataset"
species_list = north_kz_fungi_names

for folder in folders_list:
    if folder in species_list:
        current_directory = os.path.join(main_directory, folder)
        destination = os.path.join(region_directory, folder)
        copy_tree(current_directory, destination)

region_directory =
"D:\Downloads\DATA\KZ_fungi_dataset\Regional_datasets\central_kz_fungi_dataset"
species_list = central_kz_fungi_names
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
for folder in folders_list:
    if folder in species_list:
        current_directory = os.path.join(main_directory, folder)
        destination = os.path.join(region_directory, folder)
```

```
        copy_tree(current_directory, destination)

"""# 3. Проектирование модели

### Подключение необходимых пакетов
"""

# Commented out IPython magic to ensure Python compatibility.
import tensorflow as tf             #tensorflow - фреймворк для работы с ML и DL
from tensorflow import keras        #API для tensorflow

import pathlib                      #модули для работы с файлами
from glob import glob
import os
from PIL import Image
import cv2

import numpy as np                  #библиотека для обработки массивов
import pandas as pd                 #библиотека для обработки данных и их анализа
import random

from keras.preprocessing import image                           #пакеты для подготовки
файлов изображений к обработке
from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential                             #импорт модели для
создания нейронной сети с последовательной структурой слоев
from keras import layers                                        #пакет с различными
слоями для построения нейронных сетей
from tensorflow.keras.layers import Activation, Dense, Conv2D, Flatten, MaxPooling2D,
BatchNormalization

import matplotlib.pyplot as plt      #графики / визуализация
import matplotlib.image as mpimg
# %matplotlib inline

"""#### Проверка подключения и ресурсов GPU"""
!nvidia-smi

!nvcc --version # проверка наличия CUDA

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

"""## Загрузка и предобработка данных
### Загрузка датасета
"""

import splitfolders   #библиотека, позволяющая разбить данные на 3 выборки: обучающую,
валидационную и тестовую

#splitfolders.ratio(data_path, output="split_sets", seed=1337, ratio=(.8, 0.1,0.1))
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
"""#### Локальный источник данных"""

#локальный источник данных
#data_path = 'D:/Downloads/University/Diploma/Data_models/archive/Mushrooms'
```

```
data_path = 'D:/Downloads/University/Diploma/Data_models/archive/split_sets'
test_data_path =
'D:/Downloads/University/Diploma/Data_models/archive/split_sets/test_data_fungi'
train_data_path =
'D:/Downloads/University/Diploma/Data_models/archive/split_sets/train_data_fungi'
val_data_path =
'D:/Downloads/University/Diploma/Data_models/archive/split_sets/val_data_fungi'

folders_list = os.listdir(test_data_path)
folders_list

"""#### Удаленный источник данных"""
#удаленный источник данных через подключение в облаке к Google Диску
data_path =
'/content/drive/MyDrive/kz_fungi_dataset/regional_subsets_generalized/south_kz_fungi_d
ataset'

for dir in os.listdir(data_path):
  class_dir = os.path.join(data_path, dir)
  print(dir, ':', len([name for name in os.listdir(class_dir) if
os.path.isfile(os.path.join(class_dir, name))]))

#data_path = '/content/drive/MyDrive/split_sets'

test_data_path = '/content/drive/MyDrive/split_sets/test_data_fungi'
train_data_path = '/content/drive/MyDrive/split_sets/train_data_fungi'
val_data_path = '/content/drive/MyDrive/split_sets/val_data_fungi'

folders_list = os.listdir(val_data_path)
folders_list

"""#### Примеры изображений:"""

i = 0
for folder in os.listdir(train_data_path):
    image_name = random.choice(os.listdir(os.path.join(train_data_path, folder)))

    fig = plt.gcf()
    fig.set_size_inches(10, 10)
    i += 1
    axes = plt.subplot(3, 3, i)

    image = mpimg.imread(os.path.join(train_data_path, folder, image_name))
    plt.imshow(image)
    plt.title(folder)

"""### Аугментации данных и разбиение на батчи

Аугментации - это методы, которые позволяют размножить датасет путем добавления
преобразованных или измененных копий уже существующих изображений.  Подобные
преобразования включают: повороты, смещения, отражение, вращение, смена контрастности,
гаммы и т.п.
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
В Keras есть класс ImageDataGenerator, который позволяет провести аугментацию данных в
реальном времени путём генерации мини-батчей с изображениями в формате 3D тензоров.

Аугментации происходят прямо в ходе обучения каждую эпоху, что позволяет достичь
большого разнообразия данных, с аугментациями они почти не повторяются. Входные и
выходные изображения сохраняют свой размер и резрешение.
```

```
Преимущества такого способа аугментации:
*   более уникальные изображения позволяют избежать переобучение модели
*   не нужно хранить все аугментированные копии изображений, что упрощает загрузку датасета

#### Пример конфигурации аугментаций, случайно накладываемых на изображения:
"""

train_gen = ImageDataGenerator(
        rescale=1./255,                 #нормализация
        rotation_range=20,              #поворот от -20 до +20 градусов
        width_shift_range=0.2,          #смещение изображение влево/вправо не более чем на 20% всей длины
        height_shift_range=0.15,         #также, но по высоте
        shear_range=0.2,                #растягивание изображения, искажая перспективу
        zoom_range=0.2,                 #увелечение/уменьшение изображения
        #brightness_range=(0.5, 1.0),    #яркость
        horizontal_flip=True,           #отзеркаливание по вертикали
        )                               #отзеркаливание по вертикали

# изображения преобразуются в тензоры размерности 240x240x3, элементы тензора заполняются значения цветового диапазона пикселя от 0 до 255 в R/G/B цветовых каналах
# для тестовой выборки только нормализуем значения, чтобы они приняли диапазон от 0 до 1
test_gen = ImageDataGenerator(
    rescale=1./255
    )

"""#### Инициализация генераторов батчей """

from keras.applications.resnet50 import preprocess_input

image_height = 240
image_width = 240

# предобученным сетям нужно вводить свои заготовленные дата-генераторы
# preprocess_input применяется на каждое входящее изображение после преобразования размера и аугментирования (в данном случае последние не конфигурируются)
# preprocess_input переводит RGB в BGR формат и Batch Normalization на цветовые каналы
# Batch Normalization позволяет улучшить скорость сходимости градиентного спуска
data_generator = ImageDataGenerator(preprocessing_function=preprocess_input)

# загрузка и проход генератора по обучающему датасету с помощью метода ImageDataGenerator.flow_from_directory()
# из загруженных данных генератор создаст батчи случайным образом аугментированных изображений
train_generator = data_generator.flow_from_directory(
        directory=train_data_path,
        batch_size=100,                                 #размер батча
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
   target_size=(image_height, image_width),      # размер всех изображений будет транформирован в 240x240
        class_mode='categorical',                       # мультиклассовая классификация
        shuffle=True)                                   # изменение порядка изображений

batchX, batchy = train_generator.next()
print('Формат батча=%s, min=%.3f, max=%.3f' % (batchX.shape, batchX.min(),
```

```
batchX.max()))

validation_generator = data_generator.flow_from_directory(
        directory=val_data_path,
        batch_size=50,
        target_size=(image_height, image_width),
        class_mode='categorical',
        shuffle=True)

batchX, batchy = validation_generator.next()
print('Формат батча=%s, min=%.3f, max=%.3f' % (batchX.shape, batchX.min(),
batchX.max()))

test_generator = data_generator.flow_from_directory(
    directory=test_data_path,
    batch_size=1,
    target_size=(image_height, image_width),
    class_mode='categorical',
    shuffle=False)
batchX, batchy = test_generator.next()
print('Формат батча=%s, min=%.3f, max=%.3f' % (batchX.shape, batchX.min(),
batchX.max()))

train_set_size = sum(len(files) for root, dirs, files in os.walk(train_data_path))
val_set_size = sum(len(files) for root, dirs, files in os.walk(val_data_path))
test_set_size = sum(len(files) for root, dirs, files in os.walk(test_data_path))
print('Размер обучающей выборки:', train_set_size, '\nРазмер валидационной выборки:',
val_set_size, '\nРазмер тестовой выборки:', test_set_size)

train_generator.class_indices

"""#### Визуализации аугментированных изображений"""

def show_images(image):
  fig, axes = plt.subplots(1, 6, figsize=(20,20))
  axes = axes.flatten()
  for img, ax in zip(image, axes):
    ax.imshow(img)
    ax.axis('off')
  plt.tight_layout()
  plt.show()

sample_training_images, _ = next(train_generator)
show_images(sample_training_images[:6])

"""## Построение и обучение модели классификации"""

# предотвращение ошибки "Image File is truncated" при загрузки изображений PIL

from PIL import ImageFile

ImageFile.LOAD_TRUNCATED_IMAGES = True
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
"""#### Инициализация и компиляция модели

Модель проектируется по архитектуре свёрточных нейронных сетей. Sequential определяет
последовательную структуру расположения слоёв нейронной сети.

Свёрточные нейронных сети можно разделить на две части:
```

```
*   серия свёрточных слоев и pooling слоев, выделяющих признаки изображений до
абстрактных понятий
*   полносвязная нейронная сеть, выделяющая закономерности признаков

В Keras доступен пакет applications, включающий предобученные нейронные сети с
настроенными гипер-параметрами и корректированными весовыми коэфициентами

Эти модели были обучены на масштабных датасетах в течении длительного времени, поэтому
при добавлении в архитектуру своей модели их веса необходимо заморозить.

В архитектуре используется предобученная сеть ResNet50 - модель свёрточной сети от
компании Microsoft с 50 скрытыми слоями, обученная на данных ImageNet (14 млн.
помеченных разнообразных изображений, в том числе грибы). ResNet – это сокращенное
название для Residual Network

После загрузки из applications, ResNet50 устанавливается в начале архитектуры. Её
входной слой принимает 4D тензоры формата (None, 240, 240, 3) - (батч, строки,
столбцы, каналы).

На выходе выделенные свёрточными слоями 2D карты признаков будут объеденены и
преобразованы в 1D одномемерный массив, которые будет подан на вход полносвязной
нейросети.
"""

from tensorflow.keras.applications import ResNet50
model = tf.keras.Sequential()

# основа модели будет предобученная сеть архитектуры ResNet50
model.add(ResNet50(include_top = False, weights = "imagenet", pooling = "avg"))

"""В ResNet50 батчи проходят последовательные операции свёртки и батч нормализации.
Pooling слой только один, он производит объедение и сжатие после первого слоя свёртки,
тип сжатия - среднее значение элемента подвыборки"""

# построение модели с выходным слоем на 9 классов
model.add(BatchNormalization())                    # BatchNormalization()
применяется к предыдущему слою модели
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(9, activation = "softmax"))

"""1.   Скрытый слой Dense является частью полносвозвязной сети и связывает все входные
значения со своими нейронами, после чего уже все свои нейронами со всеми у следующего
слоя и применяет к ним функцию активации, в данном случае ReLu

    Параметры Dense являются гипер-параметрами всей модели, которые напрямую влияют на
процесс обучения: units - число нейронов, input_shape - число входов, activation - тип
функции активации]

    Функция промежуточных слоев ReLu: f(x) = max(0, x), с производной = 1, т.е. при
```



```
положительном аргументе она вернет сам аргумент, а при отрицательном - 0. Это
позволяет сети в равной степени корректировать веса на всех слоях сети, вместо
значительного снижения изменений к последнему слою.
2.   Выходной слой также класса Dense, число нейронов на этом слое означает число
идентифицируемых классов.

    Функция активации на выходном слое softmax - частный случай логистической функции
```

```
sigmoid для работы с векторами. softmax применяется с использованием критерия качества
categorial cross-entropy (обобщенный случай бинарной перекрестной энтропии для
классификации 3-х и более классов)
    Применение softmax очень полезно в задачах классификации, потому-что значения
выходного слоя можно интерпретировать как вероятности принадлежности внесенного
наблюдения к классам, которые привязаны к нейронам выходного слоя

Batch normalization после каждого слоя - один из способов  ускорить сходимость
алгоритма, тем самвы уменьшив время обучения, а также помогает избежать переобучения
DL моделей с большим числом скрытых слоев. Применяется также ко всем слоям ResNet50.
"""

# замораживает веса предобученной сети ResNet50
model.layers[0].trainable = False

"""**После определения структуры слоев модель компилируется**

compile создаёт модель с заданными случайными весовыми коэффициентами, после чего она
готова к обучению

Параметры компиляции (также считаются гипер-параметрами сети):

*   критерий качества (loss-функция, cost-функция) - категориальная перекрестная
энтропия, которая сравнивает распределение результатов, которые получила модель, и
верные ответы,  минимизирует градиентным спуском разницу между ними
*   способ оптимизациия градиентного спуска (поиска минимума) - Adam (настраивается
шаг сходимости/learning rate - 0.001)
Adam является одним из наиболее производительных алгоритмов оптимизации в машинном
обучении, демонстрируя высокую вычислительную эффективность. Преимущества Adam перед
другими алгоритмами:
*   свойство momentum позволяет создать схожий физическому импульс при движении
градиента в одном и том же направлении, ускоряя таким образом сходимость
*   Adam использует среднее значение вторых моментов градиентов, чтобы автоматически
динамично подбирать learning rate для отдельных параметров и адаптировать скорость
обучения.
    Например градиенты могут иногда уменьшаться на начальных слоях, из-за чего
оптимизатор повышает их LR. Т.е. для тех оптимизируемых параметров, что так получали
бы меньше корректировок, благодаря динамическому LR с Adam они получают их чаще.

Другой высоэффективный алгоритм оптимизации - это стохастический градиентный спуск
(SGD) по Нестерову, в котором также используется momentum (Nesterov momentum)

SGD по Нестерову и Adam - два основных оптимизатора, используемых при обучении
глубоких нейронных сетей.

Стартовым оптимизатором обычно выбирается Adam из-за своей возможности корректировать
скорость сходимости отдельных параметров.

SGD чаще используется, когда некоторые параметры модели уже настроены после запусков
обучения с Adam
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
*   learning rate, или же шаг сходимости - это коэффициент скорости обучения (который
можно рассматривать как шаг в правильном направлении для минимизации loss), learning
rate подбирается таким образом, чтобы он не был слишком велик и при движении не
пропускал функции, а также чтобы он не был слишком мал, т.к. это замедлит скорость
сходимости алгоритма оптимизации градиентного спуска
"""

from keras import optimizers
```

```
opt_adam = keras.optimizers.Adam(lr=0.001) #Learning rate 0.001 (определяет скорость
сходимости и обучения)
# компиляция модели
# метод оптимизации градиентного спуска - Adam
# критерий качества - категориальная перекрестная энтропия, т.к. классов больше 2
# фиксирует метрику точности классификации accuracy
model.compile(loss = 'categorical_crossentropy', optimizer = opt_adam, metrics =
['accuracy'])

#input_shape=(None, 240, 240, 3) - картинка 240 на 240 пикселя с 3 каналами цвета в
градации значений от 0 до 255
# (первый элементы кортежа отвечает за число батчей, для построения None)
input_shape = (None, 240, 240, 3)
model.build(input_shape)

#архитектура модели
model.summary()
#тип слоя ----------------- размер вывода ------ настраиваемые параметры (веса)
#resnet50 на выходе содержит Flatten слой
#Flatten преобразует изображение со свёрточных слоев в одномерный массив, который
становится входным слоем полносвязной сети
#болбшое число параметров CNN уже предварительно настроены, поэтому корректировке
подлежат только веса полносвязной сети

from keras.utils.vis_utils import plot_model

plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

"""#### Обучение модели"""

# callback - функция, которая может быть применена в ходе обучения
# такие функции могут помочь отследить информацию о параметрах модели

# данные коллбеки отслеживают точность на валидационной выборке
# EarlyStopping останавливает процесс обучения, если на протяжении 3 эпох val_accuracy
не улучшается
# restore_best_weights=False возвращает весовые коэффициенты с последней эффективной
эпохи (True - самый наивысший за все эпохи)
earlystop_cb = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', verbose=1,
                                                patience=3, restore_best_weights=False)

# ReduceLROnPlateau уменьшает значением параметра шага сходимости (learning rate) для
градиентного спуска
# меньшее значение learning rate помогает улучшить процесс оптимизации, но также ведёт
к падению скорости обучения
# если на протяжении 2 эпох val_accuracy стагнирует, то функция уменьшает
learning_rate оптимизатора на 0.01
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy',
                                                 mode='max',      #если val_accuracy
перестает расти
                                                 patience=2,      #на протяжении 2
эпох
                                                 factor=0.1,      #learning rate
уменьшается с коэффициентом 0.1 => lr*0.1
                                                 cooldown=2,      #повторно может
```

```
сработать только через 2 эпохи
                                                    min_lr=0)        #минимальное
значения learning rate


# ModelCheckpoint в конце каждой эпохи сохраняет модель и весовые коэффициенты, если
val_loss был наименьший за всех эпохи
checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(filepath='/content/sample_data',
                                                   monitor = 'val_loss', save_best_only =
True, mode = 'auto',
                                                   save_weights_only=True)
#model.load_weights(filepath)

"""для обучение применяется метод fit_generator()

в fit_generator() подаются инициализированные ранее объекты ImageDataGenerator для
обучающей и валидационной выборки, которые подают на вход сети батчи с изображениями
заданной размера. Число эпох определяет число итераций оптимизации loss функции и
обновления весовых коэффициентов

batch_size - размер подаваемое батча, т.е. в данном случае каждые 32 изображения будет
проходить коррекция весов

Размер выборки валидации - 10% от всего датасета
"""

history = model.fit_generator(train_generator, validation_data = validation_generator,
        steps_per_epoch=int(train_set_size/100) + 1,
validation_steps=int(val_set_size/50) + 1,  # steps_per_epoch * batch_size = число
образцов, обработанных в каждой эпохе
        epochs = 20, callbacks=[earlystop_cb, checkpoint_cb])
# использование многопоточности

"""Сработал EarlyStopping коллбэк и модель прекратила обучение после 14 эпох.

## Оценка качества обучения модели

### Метрики

model.evaluate() позволяет оценить качество обучения модели на валидационной или
тестовой выборке
"""

test_loss, test_accuracy = trained_model.evaluate_generator(test_generator,
steps=test_set_size)

print('Оценка точности модели на тестовой выборке:\naccuracy =', test_accuracy)

loss, accuracy = trained_model.evaluate_generator(generator=validation_generator,
steps=int(val_set_size/50) + 1)
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
print('Оценка качества модели:\naccuracy =', accuracy, '\nloss =', loss)

history_plot = pd.DataFrame(history.history)
history_plot.loc[:, ['loss', 'val_loss']].plot();
plt.title('График изменений значений функции потерь в ходе эпох обучения:')

"""Модель переобучена, о чём говорит расхождение графиков обучающей и валидационной
выборки, но тестовая выборка показывает хорошую точность."""
```

```
history_plot.loc[:, ['accuracy', 'val_accuracy']].plot();
plt.title('График изменений значений точности классификации на обучающей и валидационной выборке:')

"""### Классификатор"""

# размер тестовой выборки
filenames = test_generator.filenames
print(len(filenames))

"""Для подачи новых данных в нейронную сеть используется метод predict()

predict() использует оптимизированные весовые коэфициенты, найденные  в процессе обучения, чтобы разбирать подаваемые изображения на признаки, взвешивать их и классифицировать
"""

#model.get_weights() # получить веса модели

# перезагрузка test_generator требуется перед каждым вызовом predict_generator, чтобы сбросить предыдущие выходы
test_generator.reset()

# генерация предсказаний модели классов изображений из тестовой выборки
pred = model.predict_generator(test_generator, steps=test_set_size, verbose=1)
# выходы с выходного слоя в виде вероятностей
predicted_classes = np.argmax(pred, axis=1)                                    #
индекс элемента с максимальным значением означает номер класса

labels = test_generator.class_indices
labels_dict = dict((v,k) for k, v in labels.items())
predictions = [labels_dict[k] for k in predicted_classes]
print('Предсказанные классы:', predicted_classes)                  #результат
классификации всех изображений тестовой выборки ->
print('Идентификатор класса : ', labels)                           #-> числам
соответствуют определенные классы ->
print('Наименования по порядку: ', predictions[:10])               #-> порядкой номер
входа файла изображения имеет соответствующий индекс

# сохранение результатов классификации тестовой выборки
prediction_results = pd.DataFrame()
prediction_results["Filenames"] = filenames
prediction_results["Predictions"] = predictions
prediction_results.to_csv("prediction_results.csv", index=False)

prediction_results.head(10)

file = '/content/jpg_Il_reticolo_che_si_estende_su_tutto_il_gambo-
_Ottimo_commestibile_anche_crudo_c_Mazza.jpg'
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
img_size = 240

img = image.load_img(file, target_size=(img_size, img_size))      #загрузить своё
изображение и подгонать под нужный размер
plt.imshow(img)                                                   #показать изображение

test_image = image.img_to_array(img)                                # преобразование
изображения в 3D тензор
test_image = np.expand_dims(test_image, axis=0)                     # преобразование в
4D тензор - формат, в котором проходил обучение (за счёт батчей)
```

```
pred_p = trained_model.predict(test_image)                    # новое
предсказание
predicted_class = np.argmax(pred_p, axis=1)
                                                              # индекс элемента с
максимальным значением соответствует ->
prob_labels = test_generator.class_indices                    # -> индексу класса
изображения в test_generator
prob_labels_dict = dict((v,k) for k, v in prob_labels.items())
prob_class_prediction = [prob_labels_dict[k] for k in predicted_class]    #
наименование класса

result_prob = np.round(pred_p, 4)                              #округленные до 4-
х знаков после запятой вероятности

print('Суммы вероятностей:\n',result_prob, 'sum =', result_prob.sum(axis=1))
print('Название файла:', file.split('/')[-1])

prob_df = pd.DataFrame()                                       #вывод
вероятностей по каждому классу
prob_df['Probability'] = np.array(result_prob[0]*100)
prob_df['Species'] = prob_labels
prob_df['Edibility'] = edibility_info['edibility']
prob_df['ciryllic_name'] = edibility_info['title_rus']

title = ('Вид: %s, вероятность %.2f' % (prob_class_prediction[0],
np.amax(result_prob)*100), '%')
plt.title(title[0])

plt.savefig('prediction_example.png')
plt.show()
print('Рекоммендуется проверить предположение системы о классе, загрузив другие
фотографии образца')
prob_df

"""Данные о съедобности:"""

edibility_info =
pd.read_csv('/content/drive/MyDrive/kz_fungi_dataset/metadata/south_kz_edibility_info.
csv')
#edibility_info.drop(columns=['Unnamed: 0'])
edibility_info

"""## Сохранение и загрузка обученной модели"""

# сериализация модели в формат json
save_model = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(save_model)
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Д

```
# сериализация весовых коэффициентов в формат HDF5
model.save_weights("weights.h5")

from keras.models import model_from_json

# загрузить json и пересоздать модель
saved_model = open('/content/model.json', 'r')
recreated_model = saved_model.read()
saved_model.close()
trained_model = model_from_json(recreated_model)
```

```
# загрузить веса в модель
trained_model.load_weights("/content/weights.h5")

opt_adam = keras.optimizers.Adam(lr=0.001) #
trained_model.compile(loss = 'categorical_crossentropy', optimizer = opt_adam, metrics
= ['accuracy'])
```