

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

**ИССЛЕДОВАНИЕ МЕТОДОВ РАСПОЗНАВАНИЯ
ОБРАЗОВ В НЕЙРОСЕТЕВЫХ СИСТЕМАХ
МАШИННОГО ЗРЕНИЯ**

Студент

Митин 23.06.24
(подпись, дата)

Митин Алексей Денисович
(ФИО полностью)

Руководитель

Бождай 24.06.24
(подпись, дата)

Бождай А.С.
(фамилия, инициалы)

Нормоконтролёр

Бычков 24.06.24
(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Рецензент

Доктор Пенз. ГИ 24.06.24 Петровский С.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 23.05.24 № 11)

Заведующий кафедрой

Митрохин
(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой хороше (протокол заседания ГЭК от
25.06.2024 № 40)

Секретарь ГЭК

Г

Е.И. Гурин
(подпись)

(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 13 » 09 2022 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Митин Алексей Денисович гр. 22ВВИм2 факультета ВТ

2. Тема работы Исследование методов распознавания образов в нейросетевых системах машинного зрения

Тема утверждена приказом ПГУ № 765/0 от " 13 " 09 2022 г.

3. Руководитель магистерской диссертации д.т.н., профессор Бождай А. С.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки: исследование методов распознавания образов в нейросетевых системах машинного зрения. Разработать программное обеспечение для обучения и тестирования выбранной модели нейросетевой системы. Разработать структуру нейронной сети. Обучить нейронную сеть. В качестве демонстрации программного обеспечения использовать телеграмм-бота.

5. Перечень подлежащих разработке вопросов: 1. Анализ предметной области и методов распознавания образов в нейросетевых системах машинного зрения. 2. Разработка программного обеспечения для распознавания образов в нейросетевых системах машинного зрения. 3. Отладка программного обеспечения для распознавания образов в нейросетевых системах машинного зрения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области системы	10%	21.09.2022-21.11.2022	
2	Анализ методов распознавания	10%	22.11.2022-22.01.2023	
3	Проектирование структуры системы	20%	23.01.2023-23.04.2023	
4	Разработка всех компонентов системы	40%	24.04.2023-29.01.2024	
5	Отладка системы	10%	30.01.2024-30.03.2024	
6	Оформление выпускной работы	10%	31.03.2024-31.05.2024	

Дата выдачи задания "18" сентября 2022 г.

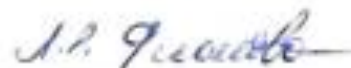
Руководитель магистерской диссертации

 
(подпись, дата)

Задание к исполнению принял студент

(подпись, дата)

Работу к защите допустить.
Декан факультета

 
24.06.2024 (подпись, дата)

РЕФЕРАТ

Магистерская работа содержит 74 с., 17 рис., 14 табл., 14 источн., 2 прил.

МАШИННОЕ ОБУЧЕНИЕ, РАСПОЗНАВАНИЕ ОБРАЗОВ, НЕЙРОСЕТЕВЫЕ СИСТЕМЫ, МАШИННОЕ ЗРЕНИЕ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

Объектом исследования являются нейросетевые системы машинного зрения.

Цель работы – повышение эффективности и точности распознавания образов с помощью использования глубоких нейронных сетей.

В ходе выполненной работы был произведен анализ предметной области, в рамках которого был произведен анализ различных методов распознавания образов в нейросетевых системах машинного зрения и составлена сравнительная таблица, которая наглядно демонстрирует различия существующих методов и определен наиболее эффективный метод, определены цели и задачи исследования методов распознавания образов в нейросетевых системах машинного зрения.

Была рассмотрена задача автоматического распознавания и категоризации изображений различных предметов одежды с использованием методов нейросетевого распознавания. Для решения этой задачи была выбрана сверточная нейронная сеть (CNN) ввиду ее эффективности, точности и способности извлекать признаки из изображений на разных уровнях абстракции, архитектура выбранной сверточной нейронной сети (CNN) для классификации изображений одежды из базы данных MNIST и работа чат бота для взаимодействия с нейросетевой системой.

ПГУ1.090401.53.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Михов А.Д.		15.06.2022
Провер.		Болжидий А.С.		16.06.2022
Н. Контр.		Болжидий А.С.		16.06.2022
Утверд.				
Исследование методов распознавания образов в нейросетевых системах машинного зрения Пояснительная записка				
Лит. Лист Листов 4 74 ФВТ, каф. ВТ гр. 22ВВм2				

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	6
ВВЕДЕНИЕ.....	7
1 Анализ предметной области	10
1.1 Обзор методов.....	10
1.2 Цель исследования	23
1.3 Описание программы	24
2 Постановка задачи	26
2.1 Основные требования.....	26
2.2 Выбор метода распознавания образа.....	27
2.3 Структурная схема системы	29
3 Архитектура и реализация ПО	35
3.1 Структура разработанной нейросети.....	35
3.2 Реализация и обучение нейросетевой модели	39
3.3 Интерфейс ПО.....	42
3.4 Тестирование ПО	43
ЗАКЛЮЧЕНИЕ	58
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	59
ПРИЛОЖЕНИЕ А Листинг программы	61
ПРИЛОЖЕНИЕ Б Презентация.....	65

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

CNN (Convolutional Neural Network) – сверточные нейронные сети

GMM (Gaussian Mixture Model) – гауссова смесь распределений

k-NN (k-Nearest Neighbors) – k-ближайших соседей

MLP (Multi-Layer Perceptron) – многослойные перцептроны

RNN (Recurrent Neural Network) – рекуррентные нейронные сети

SVM (Support Vector Machine) – метод опорных векторов

ВВЕДЕНИЕ

В современном информационном обществе развитие машинного зрения и его применение в различных областях стало неотъемлемой частью технологического прогресса. Методы распознавания образов с помощью компьютерного зрения находят широкое применение в самых различных сферах, обеспечивая повышение эффективности и автоматизацию процессов.

Одной из таких сфер является ресторанный бизнес. В ресторанах системы машинного зрения могут использоваться для проверки дресс-кода гостей. Автоматическая проверка, одеты ли посетители так, как требует правила посещения ресторана, позволяет разгрузить работников ресторана, позволяя им выделить ресурсы на другие направления своей деятельности, и повысить качество обслуживания клиентов. Камеры, оснащённые системой машинного зрения, могут распознавать тип одежды посетителей и сравнивать её с заданными критериями, что обеспечивает контроль на дресс-кодом пришедших гостей.

Технологии машинного зрения применяются в сфере безопасности и играют важную роль в распознавании и определении подозрительных личностей, таких как людей, одетых не по сезону. Это используется в местах с большим скоплением людей, таких как аэропорты, вокзалы и различные общественные мероприятия. Такие системы в режиме реального времени могут проанализировать поведение и внешний вид людей и подать сигнал на пост охраны, что позволяет оперативно реагировать на подозрительные личности и предотвращать опасные ситуации.

В сфере реклама также активно использует методы распознавания образов. С помощью машинного зрения можно анализировать внешний вид людей и определять какие бренды носят люди на улице. Такие данные дают компаниям ценную информацию для маркетинговых исследований и помогают лучше понимать предпочтения потребителей. Камеры, установленные в общественных местах, могут автоматически распознавать

логотипы и модели одежды, что позволяет собирать данные о популярности тех или иных брендов.

Однако, несмотря на эти достижения, работники в системах маркетплейсов часто сталкиваются с проблемой несоответствия названия с изображением в карточке товара. Это несоответствие может вводить в заблуждение покупателей, снижать их доверие к платформе и негативно влиять на общие продажи. Причины могут быть разнообразными: человеческий фактор при вводе данных, ошибки автоматической классификации или недостаточная точность алгоритмов распознавания образов.

Глубокие нейронные сети привлекли большое внимание в области машинного зрения благодаря своей способности автоматически извлекать высокоуровневые признаки из изображений. Сверточные нейронные сети, например, позволяют эффективно обрабатывать изображения, выделять контуры и текстуры, а также распознавать объекты на основе обученных моделей [1].

Рекуррентные нейронные сети, в свою очередь, обладают способностью анализировать последовательности данных и применяются для распознавания временных шаблонов.

В рамках этой области исследований, методы распознавания образов в нейросетевых системах машинного зрения занимают центральное положение.

Объектом исследования данной работы являются нейросетевые системы машинного зрения, а предметом исследования — методы распознавания образов в этих системах.

Мы сосредоточимся на исследовании различных подходов и алгоритмов, используемых для распознавания образов в нейросетевых системах и их применимости в реальных сценариях [2].

Актуальность темы обусловлена постоянным ростом потребности в точном и быстром распознавании образов в различных областях, таких как компьютерное зрение, автоматизация производственных процессов, медицина

и другие. Несмотря на некоторые достижения в этой области, проблема распознавания образов остается открытой и требует дальнейших исследований.

1 Анализ предметной области

1.1 Обзор методов

В настоящее время существует множество различных исследований и разработок в области методов распознавания образов в нейросетевых системах машинного зрения, как и самих методов [3]. Каждый из методов используется в зависимости от требований к нейросетевым системам, а также от задач, которые нейросеть должна решать. Ниже представлен перечень основных методов распознавания образов:

- метод опорных векторов (SVM),
- метод k-ближайших соседей (k-NN),
- гауссова смесь распределений (GMM),
- нейронные сети прямого распространения,
- рекуррентные нейронные сети (RNN),
- сверточные нейронные сети (CNN).

Рассмотрим их подробнее. Метод опорных векторов – это математический метод и статистический подход к классификации объектов. Он используется в машинном обучении как алгоритм для анализа данных и присвоения объектов определённым классам.

Границы классов, определяемые методом опорных векторов, называются классификаторами с широким зазором (Large Margin Classifier), и они оставляют вокруг границ классов как можно более широкий промежуток, свободный от объектов. SVM поддерживает как линейную, так и нелинейную классификацию. Для нелинейной классификации используется так называемый трюк с ядром (Kernel-Trick), который расширяет пространство объектов для отображения нелинейных разделительных поверхностей дополнительными измерениями (гиперплоскостями). Метод опорных векторов имеет множество областей применения. Типичное использование — распознавание образов в процессе машинного обучения компьютерных программ. Этот математический метод может применяться, например, для распознавания изображений, текста или почерка. Одним из сильных сторон

SVM является её высокая способность к обобщению для решения реальных задач.

Первые идеи о методе опорных векторов и классификации с помощью гиперплоскостей появились ещё в 1930-х годах. Основную идею метода опорных векторов разработали Владимир Вапник и Алексей Яковлевич Червоненкис много лет спустя. Прорыв SVM произошёл в 1990-х годах с увеличением использования машинного обучения. В настоящее время существует множество адаптаций и дальнейших разработок SVM.

Одна из центральных задач машинного обучения — это классификация данных. Цель классификации — на основе имеющихся данных и их принадлежности к определённым классам определить, к какому классу можно отнести новый объект данных [4]. Для начала требуется база данных с обучающими объектами, чья принадлежность к классам известна. На основе этих данных различные алгоритмы машинного обучения пытаются найти разделяющие линии или поверхности, с помощью которых новые объекты можно будет отнести к правильным классам.

Метод опорных векторов (SVM) делит классы объектов с помощью гиперплоскостей. Эти гиперплоскости выбираются таким образом, чтобы между различными классами оставался как можно более широкий промежуток, свободный от объектов. Разделительная поверхность с наибольшим свободным от объектов промежутком считается оптимальным решением. SVM поддерживает как линейную, так и нелинейную классификацию объектов. В реальных задачах данные объекты в большинстве случаев не могут быть классифицированы с помощью чисто линейных границ. Для нелинейных границ классов метод опорных векторов использует так называемый трюк с ядром.

При этом объекты переводятся в пространство более высокой размерности (гиперплоскость). Чем выше количество измерений, тем более сложные разделительные поверхности можно реализовать с помощью гиперплоскостей. Преобразование линейных разделительных поверхностей

различных гиперплоскостей в нелинейные разделительные поверхности происходит при обратном преобразовании измерений. Могут возникнуть даже несвязные разделительные поверхности.

Чтобы преобразование объектов в пространство высокой размерности и обратное преобразование не было слишком вычислительно затратным, SVM использует так называемые функции ядра для описания разделительных поверхностей. Термин «трюк с ядром» происходит от использования этих функций ядра.

Для метода опорных векторов существует множество применений для решения реальных задач с помощью машинного обучения. Часто встречающееся применение — распознавание почерка. На основе обучающих данных SVM может определить подходящие разделительные поверхности, с помощью которых рукописные символы можно надёжно отнести к соответствующим буквам или цифрам. Другой областью применения является распознавание текста (оптическое распознавание символов — OCR). Также с помощью метода опорных векторов можно достичь хороших результатов в распознавании изображений [5].

Метод k-NN (k-Nearest Neighbor) — это основанный на близости алгоритм машинного обучения, который сравнивает точки данных с набором данных, на котором он был обучен и который хранит для предсказаний. Благодаря этой функции инстанционного обучения k-NN является «ленивым» алгоритмом, который хорошо подходит для задач классификации или регрессии. Алгоритм k-NN основывается на предположении, что похожие точки находятся близко друг к другу — подобное притягивает подобное.

Как алгоритм классификации, k-NN присваивает новым точкам данных категорию, которая наиболее распространена среди их ближайших соседей. Как алгоритм регрессии, k-NN делает предсказания на основе среднего значения точек, находящихся рядом с точкой запроса. k-NN является алгоритмом с учителем, где «k» обозначает количество ближайших соседей,

которые учитываются для задачи классификации или регрессии, в то время как «NN» обозначает ближайших соседей выбранного числа «k» [6].

k-NN был изначально разработан в 1951 году Эвелин Фикс и Джозефом Ходжесом в ходе их исследовательской работы для армии США. Они опубликовали описание метода дисперсионного анализа, являющегося непараметрическим методом классификации. В 1967 году Томас Ковер и Питер Харт расширили этот непараметрический метод классификации и опубликовали свою статью «Nearest Neighbor Pattern Classification». Почти 20 лет спустя алгоритм был усовершенствован Джеймсом Келлером до «Fuzzy k-NN» с целью снижения уровня ошибок.

Сегодня алгоритм k-NN широко используется благодаря своей адаптивности к различным областям — от генетики и финансов до обслуживания клиентов. Алгоритм k-NN является алгоритмом с учителем, что означает, что он обучается на тренировочных данных и запоминает их. На основе этих данных алгоритм создаёт функцию, которая выдаёт подходящий результат для новых, неразмеченных данных.

Таким образом, алгоритм может решать задачи классификации или регрессии. Хотя вычисления k-NN выполняются во время запроса, а не на этапе обучения, алгоритм предъявляет значительные требования к памяти и поэтому довольно ресурсоёмок [7]. Для задач классификации алгоритм k-NN присваивает метку класса на основе большинства голосов и использует метку, которая наиболее часто встречается вокруг заданной точки данных. Выход классификационной задачи — это мода ближайших соседей.

Для задач регрессии k-NN использует среднее значение ближайших соседей для предсказания числового значения. Проблемы регрессии возвращают числовые значения в качестве результата запроса [8]. Например, если вы создаете диаграмму для предсказания веса человека на основе его роста, то значения роста являются независимой переменной, а значения веса — зависимой переменной. На основе среднего соотношения между ростом и

весом можно оценить вес человека (зависимая переменная) на основе его роста (независимая переменная).

Ключом к алгоритму k-NN является определение расстояния между точкой запроса и другими точками данных. С помощью метрик расстояния можно установить границы принятия решений. Эти границы формируют различные области данных. Существуют разные методы расчета расстояний. В основном используются методы: Евклидово расстояние, Манхэттенское расстояние, Расстояние Минковского, Расстояние Хэмминга.

Евклидово расстояние — это наиболее часто используемая мера расстояния, которая использует прямую линию между точкой запроса и другой измеряемой точкой.

Манхэттенское расстояние — это также популярная мера расстояния, которая вычисляет абсолютную разницу между двумя точками. Оно представляется на сетке и иногда называется «такси»-геометрией: как добраться от точки А (точки запроса) до точки В (измеряемой точки)?

Расстояние Минковского — это обобщение евклидовой и манхэттенской метрик расстояния, которое позволяет создавать новые метрики расстояния [9]. Оно рассчитывается в нормированном векторном пространстве. В расстоянии Минковского параметр "p" определяет используемое при расчете расстояние. При $p=1$ используется манхэттенское расстояние. При $p=2$ используется евклидово расстояние.

Расстояние Хэмминга — иногда называется метрикой совпадения и использует булевы или строковые векторы для определения, где векторы не совпадают. Оно измеряет расстояние между двумя строками одинаковой длины. Это особенно полезно для обнаружения и исправления ошибок в кодах.

Алгоритм k-NN использует эти метрики для определения ближайших соседей, что позволяет эффективно решать задачи классификации и регрессии, основываясь на анализе данных. Алгоритм k-NN, известный своей простотой и точностью, находит применение во многих областях, особенно в анализе классификации: ранжирование по релевантности, поиск сходства

изображений или видео, распознавание образов, финансы, рекомендации по продуктам и механизмы рекомендаций, здравоохранение, предварительная обработка данных.

GMM — это параметрическая плотность вероятности, представленная в виде взвешенной суммы гауссовских компонентных плотностей. GMM часто используется в качестве параметрической модели вероятностного распределения непрерывных измерений или признаков в биометрической системе, таких как спектральные признаки, связанные с голосовым трактом в системе распознавания говорящего. Параметры GMM оцениваются на основе обучающих данных с использованием итеративного алгоритма ожидания-максимизации (ЕМ) или оценки максимального апостериорного правдоподобия (МАР) на основе хорошо обученной предварительной модели. GMM используются в различных приложениях машинного обучения, включая кластеризацию, оценку плотности и распознавание образов.

Модель смеси — это вероятностная модель для представления данных, которые могут поступать из нескольких разных источников или категорий, каждая из которых моделируется отдельным распределением вероятностей. Например, финансовая доходность обычно ведет себя по-разному в нормальных рыночных условиях и в периоды кризиса, и поэтому ее можно смоделировать как смесь двух различных распределений. Модель смеси предполагает, что каждая точка данных происходит из одного из распределений компонентов, при этом конкретное распределение выбирается в соответствии с весами смешивания.

Модель не требует знания того, какому компоненту принадлежит каждая точка данных. В отличие от k -средних, которые предполагают сферические кластеры, GMM могут адаптироваться к эллипсоидным формам благодаря ковариационному компоненту [10]. Это позволяет GMM захватывать более широкий спектр форм кластеров.

Также они могут работать с кластерами разного размера благодаря использованию ковариационных матриц и коэффициентов смешивания,

которые учитывают разброс и долю каждого кластера, а также могут работать с перекрывающимися кластерами, поскольку точки данных распределяются по кластерам на основе вероятностей, а не жестких границ. GMM предоставляют вероятности (мягкие назначения) каждой точки, принадлежащей каждому кластеру, что может быть более информативным для понимания данных. Результаты кластеризации легко объяснить, поскольку каждый кластер представлен распределением Гаусса с определенными параметрами. Помимо кластеризации, GMM также можно использовать для оценки плотности и обнаружения аномалий.

Нейронные сети прямого распространения, также известные как многослойные перцептроны (MLP), состоят из входного слоя, одного или нескольких скрытых слоев и выходного слоя.

Они используют алгоритм прямого распространения для обработки входных данных и генерации прогнозов на выходе. MLP широко используются в различных областях, включая, среди прочего, распознавание изображений, обработку естественного языка и распознавание речи. Их гибкость в архитектуре и способность аппроксимировать любую функцию в заданных условиях делают их фундаментальным строительным блоком в глубоком обучении и исследованиях нейронных сетей. Давайте углубимся в некоторые из его основных концепций.

Входной слой состоит из узлов или нейронов, которые получают исходные входные данные. Каждый нейрон представляет собой функцию или измерение входных данных. Количество нейронов во входном слое определяется размерностью входных данных. Нейроны соседних слоев полностью связаны друг с другом. Каждому соединению соответствует вес, который определяет прочность соединения. Эти веса изучаются в процессе тренировки. Помимо входных и скрытых нейронов, каждый слой (кроме входного слоя) обычно включает в себя нейрон смещения, который обеспечивает постоянный входной сигнал для нейронов следующего слоя.

Нейрон смещения имеет свой вес, связанный с каждым соединением, который также изучается во время обучения.

Нейрон смещения эффективно меняет функцию активации нейронов на следующем слое, позволяя сети изучить смещение или смещение на границе принятия решения. Регулируя веса, связанные с нейроном смещения, MLP может научиться контролировать порог активации и лучше соответствовать обучающим данным. Обычно каждый нейрон в скрытых слоях и выходном слое применяет функцию активации к своей взвешенной сумме входных данных. Общие функции активации включают сигмоидальную, танх, ReLU (выпрямленную линейную единицу) и softmax. Эти функции вносят в сеть нелинейность, позволяя ей изучать сложные закономерности в данных. MLP обучаются с использованием алгоритма обратного распространения ошибки, который вычисляет градиенты функции потерь относительно параметров модели и итеративно обновляет параметры, чтобы минимизировать потери.

RNN предназначены для моделирования последовательных данных путем использования обратной связи. Они имеют ячейки памяти, которые позволяют сохранять информацию во времени, что делает их подходящими для задач, таких как распознавание речи, обработка естественного языка и анализ временных рядов [11]. Первой искусственной нейронной сетью такого рода стала сеть Джордана, в которой каждая скрытая ячейка получала собственный вывод с фиксированной задержкой — одной или несколькими итерациями. В остальном это сравнимо с классическими сетями прямой связи.

Конечно, существует множество вариаций — например, передача состояния входным узлам, переменные задержки и т. д., но основная идея остается той же. Этот тип нейронных сетей особенно используется, когда важен контекст. Потому что тогда решения прошлых итераций или выборов существенно влияют на текущие.

Однако, поскольку у рекуррентных сетей есть существенный недостаток: они со временем становятся нестабильными, в настоящее время обычной практикой является использование так называемых блоков

долговременной краткосрочной памяти (сокращенно LSTM). Они также стабилизируют RNN для зависимостей, существующих в течение длительного периода времени.

Наиболее распространенным примером таких отношений является обработка текста: слово можно анализировать только в контексте предыдущих слов или предложений. Другой пример — обработка видео, например, при автономном вождении. Объекты в последовательности изображений распознаются и отслеживаются с течением времени.

CNN представляет собой специализированный тип алгоритма глубокого обучения, предназначенный в первую очередь для задач, требующих распознавания объектов, таких как классификация, обнаружение и сегментация изображений [12]. CNN используются в различных случаях, например, в автономных транспортных средствах, системах камер видеонаблюдения и других. Сверточная нейронная сеть состоит из четырех основных частей.

Сверточные слои – первый компонент CNN. Как следует из названия, основная выполняемая математическая задача называется сверткой, которая представляет собой применение функции скользящего окна к массиву пикселей, представляющих изображение. Функция скольжения, применяемая к массиву, называется ядром или фильтром, и оба названия могут использоваться как взаимозаменяемые [13].

В сверточном слое применяется несколько фильтров одинакового размера, и каждый фильтр используется для распознавания определенного шаблона на изображении, например, кривизны цифр, краев, всей формы цифр и т. д. Проще говоря, в сверточном слое мы используем небольшие сетки (называемые фильтрами или ядрами), которые перемещаются по изображению.

Каждая маленькая сетка похожа на мини-увеличительное стекло, которое ищет на фотографии определенные узоры, такие как линии, кривые или формы. Когда вы прокручиваете фотографию, создается новая сетка,

которая выделяет участки, где были найдены соответствующие узоры. Например, один фильтр может быть эффективен для обнаружения прямых линий, другой — для поиска изогнутых форм и т. д. Применяя несколько различных фильтров, сверточная нейронная сеть (CNN) способна получить полное представление обо всех разнообразных шаблонах, присутствующих на изображении.

Функция активации ReLU (Rectified Linear Unit) используется после каждой операции свертки. Эта функция способствует обучению сети путем выявления нелинейных зависимостей между элементами изображения, что повышает ее способность распознавать различные закономерности. Кроме того, ReLU помогает уменьшить проблему затухания градиента, улучшая стабильность и эффективность обучения.

Слои пулинга (pooling) играют ключевую роль в извлечении наиболее значимых признаков из свернутой матрицы. Это достигается посредством применения операций агрегации, которые уменьшают размерность карты признаков (свернутой матрицы), тем самым снижая объем памяти, используемой во время обучения сети. Пулинг также важен для предотвращения переобучения, улучшая обобщающую способность модели.

Полносвязные слои (fully connected layers) расположены в завершающей части сверточной нейронной сети. Они принимают на вход сглаженную одномерную матрицу, сформированную предыдущими слоями пулинга. В этих слоях также применяются функции активации ReLU для придания нелинейности.

Сверточные нейронные сети произвели революцию в области компьютерного зрения, обеспечив значительные достижения в ряде реальных приложений, таких как классификация изображений, обнаружение объектов и распознавание лиц.

Для удобства сравнения и анализа в таблице 1 представлено сопоставление этих методов по определенным критериям, что помогает определить наиболее подходящий подход для конкретных задач.

Таблица 1 – Сравнение методов

Критерии	SVM	kNN	GMM	MLP	RNN	CNN
Хорошая обобщающая способность	+		+	+	+	+
Эффективность на малых обучающих выборках	+	+	+			
Может работать с линейно неразделимыми классами данных	+		+	+	+	+
Простота реализации и понимания	+	+				
Гибкость в обработке нелинейных данных			+	+	+	+
Не требует обучения		+				
Гибкость моделирования различных форм распределений			+	+	+	+
Устойчивость к шуму и выбросам	+			+	+	+
Автоматическое выделение признаков						+
Подходит для обработки изображений						+
Высокая производительность на больших данных				+	+	+
Может работать с неоднородными кластерами данных			+		+	+
Возможность параллельной обработки	+			+	+	+

Данные критерии были выбраны для сравнения методов в рамках производительности, удобства и гибкости в использования. Следует подчеркнуть, что эти критерии являются общепринятыми для сравнения методов в области нейросетевых систем и машинного обучения, а также они являются отражающими основные характеристики при выборе методов под

конкретную задачу. Для большего понимания значимости выбранных критериев их определение было приведено в таблице 2.

Таблица 2 – Определение критериев

Критерии	Определение
Хорошая обобщающая способность	Способность модели хорошо работать на новых, ранее невиданных данных
Эффективность на малых обучающих выборках	Способность модели хорошо работать даже с небольшим объемом обучающих данных
Может работать с линейно неразделимыми классами данных	Способность модели работать с данными, которые нельзя разделить линейно
Простота реализации и понимания	Насколько просто реализовать и понять модель
Гибкость в обработке нелинейных данных	Способность модели работать с нелинейными зависимостями в данных
Не требует обучения	Возможность модели работать без обучения
Гибкость моделирования различных форм распределений	Способность модели работать с различными распределениями данных
Устойчивость к шуму и выбросам	Насколько хорошо модель справляется с шумными данными и выбросами
Автоматическое выделение признаков	Способность модели автоматически выделять важные признаки из данных

Продолжение таблицы 2

Критерии	Определение
Подходит для обработки изображений	Насколько хорошо модель подходит для обработки и анализа изображений
Высокая производительность на больших данных	Способность модели эффективно работать с большими объемами данных
Может работать с неоднородными кластерами данных	Способность модели работать с неоднородными и сложными кластерами данных расширяет её применение для различных типов данных и задач
Возможность параллельной обработки	Возможность параллельной обработки улучшает производительность модели, особенно при работе с большими объемами данных или сложными вычислениями

Таким образом были выбраны критерии для всесторонней оценки методов машинного обучения, составлена сравнительная таблица и проведен анализ выбранных методов. Таблица наглядно демонстрирует различия существующих методов, на основе которых сверточные нейронные сети (CNN) выделяются на общем фоне как наиболее подходящий по многим из критериев.

В последние годы наблюдается стремительный рост объемов данных и информации, требующих автоматической обработки и анализа. Машинное зрение, как одна из важнейших областей искусственного интеллекта, позволяет решать задачи, связанные с интерпретацией и пониманием

визуальной информации. В маркетплейсах, где ежедневно обрабатываются тысячи изображений товаров, корректность и точность распознавания образов играют критически важную роль. Ошибки в сопоставлении названий товаров с их изображениями могут привести к недовольству покупателей, снижению доверия к платформе и финансовым потерям. Это подчеркивает актуальность исследования и необходимость разработки более точных и надежных методов распознавания образов.

1.2 Цель исследования

Данная работа направлена на решение ряда ключевых задач в области методов распознавания образов в нейросетевых системах машинного зрения. Целью исследования является повышение эффективности и точности распознавания образов с использованием глубоких нейронных сетей, с особым акцентом на проблемы, возникающие в маркетплейсах, такие как несоответствие названий товаров с их изображениями. Для достижения данной цели были поставлены следующие конкретные задачи:

- провести обзор и систематизацию существующих методов классификации одежды с использованием нейросетевых систем, включая SVM, k-NN, GMM, MLP, RNN и CNN, и оценить их преимущества и недостатки,
- выбрать наиболее подходящую модель нейросетевой системы для решения конкретных задач распознавания образов,
- разработать программное обеспечение для обучения и тестирования выбранной модели нейросетевой системы,
- провести эксперименты для оценки производительности выбранной модели в различных сценариях и условиях классификации одежды, включая разнообразные типы одежды.

После решения данных задач предполагается получить понимание возможностей нейросетевых систем машинного зрения и их ограничений в рамках задач распознавания образов, а также предложить практические рекомендации по оптимизации и улучшения их работы.

1.3 Описание программы

На основании вышесказанного можно сделать вывод, что для решения поставленных задач должна быть разработана программная система, которая будет обучать и тестировать нейросетевую систему машинного зрения. Программа должна быть удобной и эффективной, построенной с использованием современных библиотек и фреймворков для глубокого обучения. При реализации должна быть создана с учетом принципов модульности и расширяемости, что позволит легко внедрять изменения в архитектуру модели, а также добавить новые функциональности в будущем.

Для создания нейросетевой модели должен использоваться современный и популярный фреймворк глубокого обучения с высокой производительностью и поддержкой различных архитектур сетей.

Разработанная нейросетевая модель должна быть основана на сверточной нейронной сети (CNN), настроенной для распознавания образов. Модель должна включать в себя входной слой, сверточные слои для выделения признаков, слои пулинга для уменьшения размерности, полносвязные слои и слой вывода.

Программа должна предоставлять возможность обучения нейросетевой модели на предоставленных обучающих данных. Также должны использоваться передовые методы оптимизации и обратного распространения ошибки для обучения модели на основе предоставленных образцов.

Разработанная программа должна включать в себя модуль для тестирования и валидации нейросетевой модели на независимых тестовых данных.

Должен быть разработан графический интерфейс пользователя для удобства использования программы, с возможностью загрузки данных, настройки параметров обучения, мониторинга процесса обучения и анализа результатов тестирования.

Разработанная программа должна позволять экспортировать обученную нейросетевую модель для последующего использования в реальных

приложениях. Модель должна иметь возможность сохранения в форматах, совместимых с различными платформами и технологиями.

Входные данные должны быть представлены в формате изображений с образцами для распознавания. Выходные данные должны быть представлены в виде результата классификации изображения.

Система должна обладать следующим функционалом:

- загрузка и обработка изображений, полученные из файлов,
- выполнение предварительной обработки изображений, а именно изменение размера, нормализацию или фильтрацию шума,
- обучение нейросетевых моделей на основе обучающей выборки для достижения высокой точности распознавания образов,
- распознавание образов на новых изображениях с высокой скоростью и точностью.

2 Постановка задачи

2.1 Основные требования

Основными требованиями, предъявляемые к нейросетевой системе, являются:

- изображения должны иметь фиксированное разрешение в размере 28x28 пикселей и должны быть представлены в черно-белом формате (градации серого),
- система должна классифицировать изображения на 10 классов,
- изображения должны быть представлены в виде матрицы пикселей, где каждый пиксель имеет значение от 0 до 255, что определяет интенсивность серого цвета,
- для обучения и тестирования системы должны быть использован стандартный набор данных Fashion MNIST, содержащий около 60 тыс. обучающих и 10,000 тестовых изображений различных предметов одежды.

Классы для классификации:

- футболка/топ (t-shirt/top),
- брюки (trouser),
- свитер (pullover),
- платье (dress),
- пальто (coat),
- сандалии (sandal),
- рубашка (shirt),
- кроссовки (sneaker),
- сумка (bag),
- ботинки (ankle boot).

Особенности распознаваемых изображений:

- изображения представляют собой чёрно-белые снимки различных предметов одежды,

- каждое изображение имеет фиксированный размер 28x28 пикселей, что обеспечивает унификацию входных данных для нейросети,

- в изображениях могут присутствовать незначительные шумы и искажения, связанные с различными условиями съемки (освещение, фон, ракурс и т.д.),

- вариативность включает разные стили одежды, оттенки серого цвета и мелкие детали, что требует от нейросети способности к обобщению и точному распознаванию.

Ожидаемые результаты от реализованной системы:

- каждое входное изображение должно быть классифицировано в одну из 10 категорий одежды, указанных выше,

- для каждого изображения система должна предоставлять вероятностное распределение по всем классам, где наибольшая вероятность указывает на предсказанную категорию,

- система должна генерировать отчеты о точности классификации на обучающих и тестовых наборах данных для оценки эффективности модели.

2.2 Выбор метода распознавания образа

На сегодняшний день существует множество методов нейросетевого распознавания, включая сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN), глубокие нейронные сети прямого распространения (DNN) и многослойные перцептроны (MLP). Каждый из этих методов имеет свои уникальные особенности и области применения. CNN особенно эффективны в работе с изображениями, поскольку они способны автоматически извлекать иерархические признаки из данных. Они нашли широкое применение в таких областях, как компьютерное зрение, распознавание объектов и обработка изображений. RNN хорошо подходят для обработки последовательных данных, таких как текст, речь и временные ряды. Они способны учитывать контекст при анализе данных, что делает их эффективными для задач, таких как машинное перевод, распознавание речи и прогнозирование временных рядов. DNN представляют собой многослойные

перцептроны с большим количеством скрытых слоев. Они способны обучаться на больших объемах данных и решать сложные задачи, такие как классификация изображений, распознавание речи и обработка естественного языка. MLP - это простейшие нейронные сети, состоящие из одного или нескольких скрытых слоев. Они используются для решения задач классификации и прогнозирования, где данные представлены в виде фиксированных векторов.

Сверточные нейронные сети (CNN) стали стандартным методом для классификации изображений благодаря своей эффективности и точности. Они хорошо подходят для обработки пространственных данных, таких как изображения, благодаря использованию операций свертки и пулинга.

Сверточные нейронные сети (CNN) обладают определенными преимуществами, выделяющие их на фоне остальных нейронных сетей, а именно:

- CNN обладают способностью автоматически извлекать признаки из изображений на разных уровнях абстракции,
- CNN демонстрируют высокую обобщающую способность,
- CNN, как правило, требуют меньше параметров, чем полносвязанные нейронные сети,
- CNN достигают высокой точности в задачах классификации изображений.

Рассмотрим вышеописанные преимущества подробнее. CNN обладают способностью автоматически извлекать признаки из изображений на разных уровнях абстракции. Это достигается за счет использования сверточных и подвыборочных слоев, которые позволяют сети фокусироваться на локальных паттернах и постепенно переходить к более глобальным характеристикам изображения. Благодаря такой архитектуре CNN способны эффективно распознавать объекты на изображениях, даже если они представлены с разных ракурсов, в различных условиях освещения или с некоторыми искажениями.

CNN демонстрируют высокую обобщающую способность, что означает, что они могут успешно распознавать объекты на изображениях, даже если они немного отличаются от тех, что были использованы в процессе обучения. Это свойство обуславливается использованием сверточных операций, которые позволяют сети инвариантно относиться к некоторым вариациям изображения, сохраняя при этом его ключевые характеристики.

CNN, как правило, требуют меньше параметров, чем полносвязанные нейронные сети, что делает их более эффективными в плане вычислений и использования памяти. Это преимущество особенно важно при работе с задачами, где требуется обработка больших объемов изображений, например, при классификации изображений одежды из интернет-магазинов или при распознавании объектов на видеопотоках.

CNN достигают высокой точности в задачах классификации изображений, что делает их привлекательным решением для различных приложений, включая классификацию одежды. Многочисленные исследования показали, что CNN способны превосходить другие методы машинного обучения, такие как SVM (Support Vector Machines) и K-Nearest Neighbors (KNN), в задачах классификации изображений одежды.

В данном методе есть не только преимущества, но и недостатки:

- метод CNN требуют большого количества размеченных данных для эффективного обучения и извлечения признаков,
- обучение CNN может требовать значительного времени и ресурсов вычислений, особенно при использовании глубоких архитектур,
- в некоторых случаях сложно интерпретировать, какие признаки именно извлекает CNN для классификации изображений, что может затруднять анализ и диагностику моделей.

2.3 Структурная схема системы

В ходе разработки данной системы была разработана структурная схема системы. Данная схема отображает основные элементы разработанной

системы и взаимосвязи данных элементов. В разработанной системе можно выделить следующие основные блоки:

- telegram бот,
- модуль автоматизации подачи данных и представления результатов,
- модуль предварительной обработки изображений,
- нейросетевая система для распознавания образов.

Структурная схема разработанной системы представлена ниже на рисунке 1.

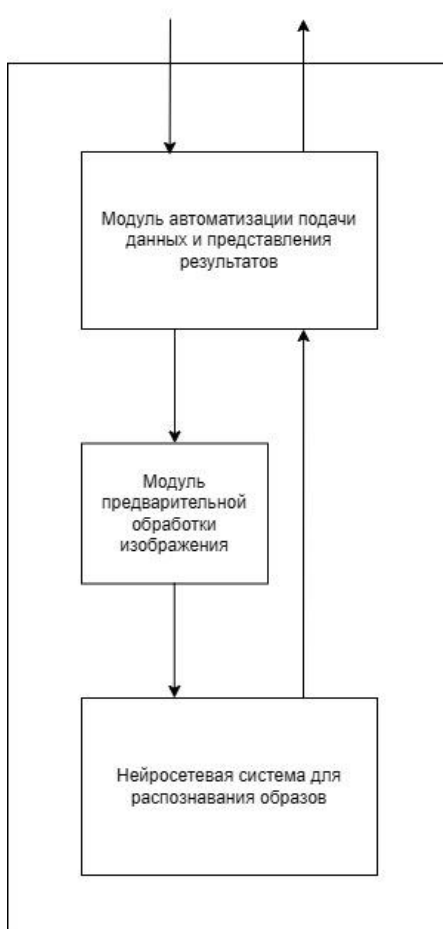


Рисунок 1 – Структурная схема системы

Для отображения взаимодействия компонентов системы между собой были построены диаграммы последовательностей. Данные диаграммы используются для демонстрации всего жизненного цикла объекта и взаимодействия между объектами в порядке, в котором данные взаимодействия происходят. Определение очередности событий для

достижения определенного результата – это является главной задачей диаграммы последовательности.

В результате работы были построены следующие диаграммы:

- диаграмма последовательности для классификации изображения,
- диаграмма последовательности для обработки ошибки и повторной попытки распознавания,
- диаграмма последовательности для классификации нескольких изображений.

Данные диаграммы позволяют наглядно представить взаимодействие компонентов в рамках разработанной системы, а также увидеть последовательность выполнения операций. Рассмотрим построенные диаграммы подробнее.

Диаграмма последовательности для классификации изображения представлена на рисунке 2.

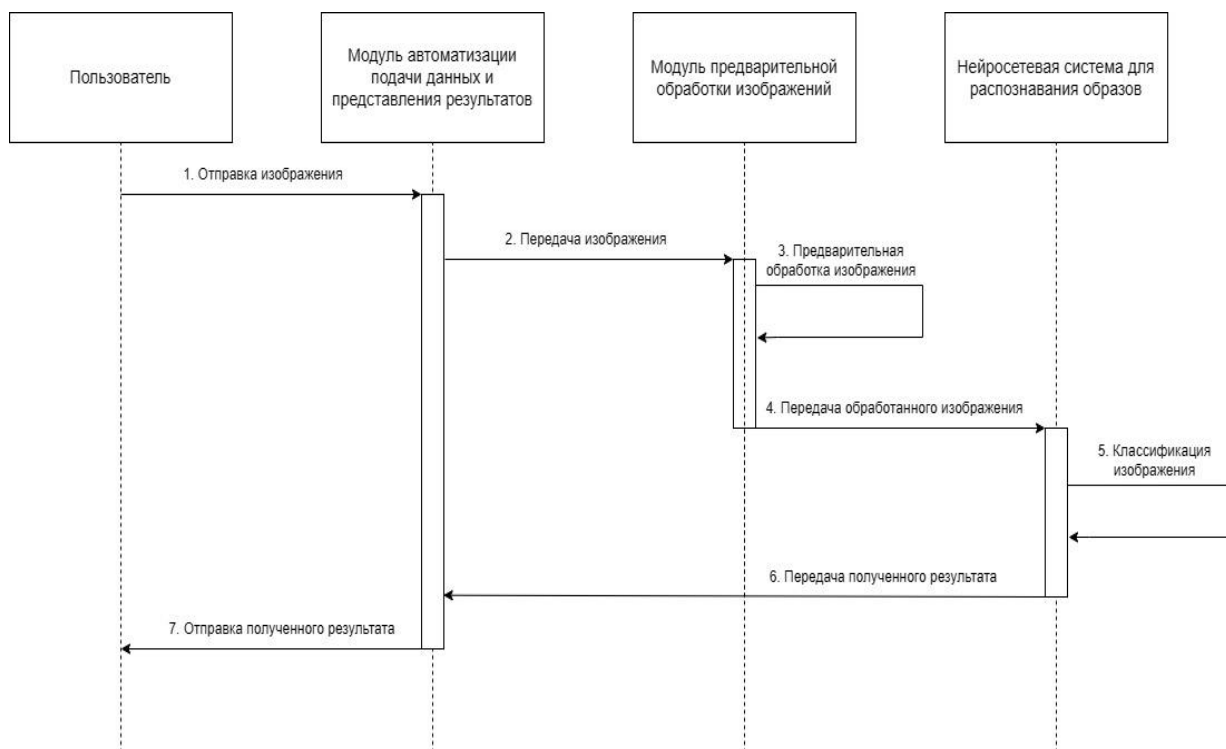


Рисунок 2 – Диаграмма последовательности для классификации изображения

Пользователь начинает взаимодействие с Telegram ботом, отправляя изображение для последующего распознавания. Бот получает переданные

данные и направляет их в модуль предварительной обработки изображений, где происходит изменение размера, нормализация и другие подготовительные операции. После этого обработанное изображение передается в нейросетевую систему для выполнения задачи распознавания образов.

Нейросетевая система анализирует обработанное изображение и формирует результаты распознавания, которые затем передаются модулю автоматизации. Этот модуль отвечает за управление передачей данных между компонентами системы и автоматическую отправку результатов обратно пользователю через Telegram бота.

После успешного завершения процесса распознавания и получения результатов, Телеграмм-бот уведомляет пользователя о готовности данных. Пользователь получает результаты распознавания через бота, что является завершающей точкой в этом цикле обработки изображения.

Диаграмма последовательности для обработки ошибки и повторной попытки распознавания представлена на рисунке 3.

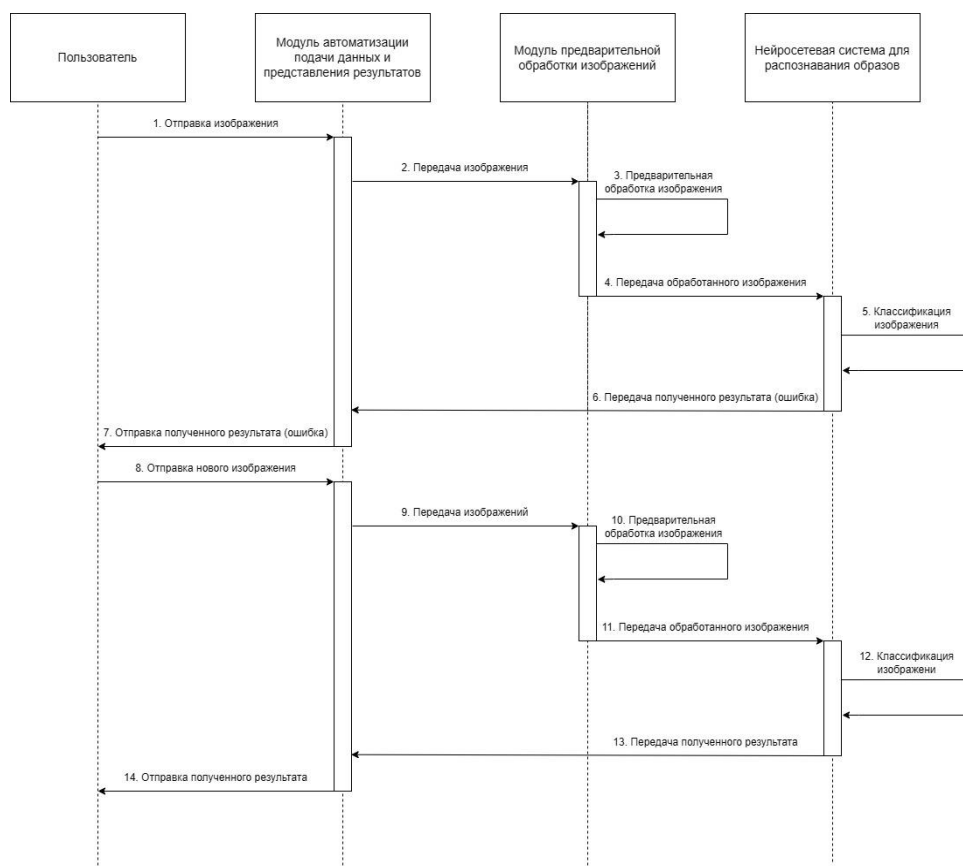


Рисунок 3 – Диаграмма последовательности для обработки ошибки и повторной попытки распознавания

Пользователь начинает взаимодействие с ботом с помощью мессенджера Telegram, отправляя изображение для распознавания. Бот получает изображение и направляет его в модуль предварительной обработки, где происходит изменение размера, нормализация и другие подготовительные операции. Затем обработанное изображение передается в нейросетевую систему для распознавания.

Если в процессе распознавания возникает ошибка, нейросетевая система передает информацию об ошибке в модуль автоматизации. Этот модуль отправляет уведомление об ошибке пользователю через Telegram бота, запрашивая повторную отправку изображения.

Пользователь повторно отправляет изображение через бота. Бот снова направляет изображение в модуль предварительной обработки для необходимых подготовительных операций. После этого обработанное изображение повторно передается в нейросетевую систему для распознавания.

Нейросетевая система анализирует изображение и формирует результаты распознавания, которые снова передаются в модуль автоматизации. Этот модуль управляет передачей данных между компонентами системы и отправляет результаты пользователю через Telegram бота. Пользователь получает результаты распознавания, завершая взаимодействие в этом цикле обработки изображений.

Диаграмма последовательности для классификации нескольких изображений представлена на рисунке 4.

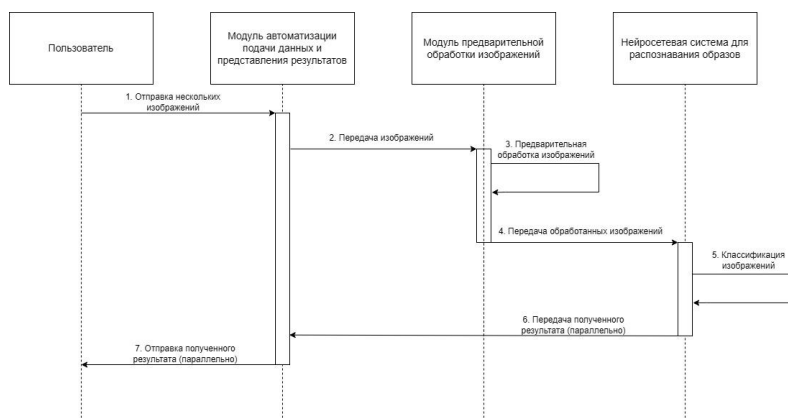


Рисунок 4 – Диаграмма последовательности для классификации нескольких изображений

Пользователь начинает взаимодействие с Telegram ботом, отправляя изображение для последующего распознавания. Бот получает переданные данные и направляет их в модуль предварительной обработки изображений, где происходит изменение размера, нормализация и другие подготовительные операции. После этого обработанное изображение передается в нейросетевую систему для выполнения задачи распознавания образов.

Нейросетевая система анализирует обработанное изображение и формирует результаты распознавания, которые затем передаются модулю автоматизации. Этот модуль отвечает за управление передачей данных между компонентами системы и автоматическую отправку результатов обратно пользователю через Telegram бота.

После успешного завершения процесса распознавания и получения результатов, Телеграмм-бот уведомляет пользователя о готовности данных. Пользователь получает результаты распознавания через бота, что является завершающей точкой в этом цикле обработки изображения.

3 Архитектура и реализация ПО

В данной работе мы рассмотрим и реализуем архитектуру выбранной сверточной нейронной сети (CNN) для классификации изображений одежды из базы данных MNIST. Учитывая специфику этой базы данных, мы подробно опишем архитектуру и реализацию нашего ПО.

3.1 Структура разработанной нейросети

В рамках данной работы была разработана нейросетевая система с определенной архитектурой, включающей девять основных элементов:

- входное изображение,
- первый сверточный слой,
- второй сверточный слой,
- первый слой подвыборки,
- третий сверточный слой,
- четвертый сверточный слой,
- второй слой подвыборки,
- полносвязный слой,
- выходной слой.

Разработанная сверточная сеть состоит из двух повторяющихся последовательных участков. Каждый такой участок включает в себя два сверточных слоя и слой подвыборки, предназначенные для выделения признаков изображения. В заключительной части сети расположен классификатор, состоящий из полносвязного слоя с 512 нейронами и выходного слоя, включающего 10 нейронов. Эти нейроны содержат вероятности принадлежности изображений к тому или иному классу, что является ключевым для корректной классификации входных данных.

На вход нейронной сети подаются изображения размером 32×32 пикселя с тремя каналами: красным, зеленым и синим. Первый сверточный слой содержит 32 карты признаков размером 3×3 , то есть каждый нейрон сверточного слоя подключен к квадратному участку изображения размером 3×3 . Второй сверточный слой имеет аналогичную архитектуру, также

состоящую из 32 карт признаков с ядром свертки 3×3 . После этого слоя следует первый слой подвыборки, который уменьшает размерность каждого из признаков. Этот слой также содержит 32 карты признаков, а размер поля выборки составляет 2×2 .

На вход нейронной сети подаются изображения размером 32×32 пикселя, представленные в трех каналах: красном, зеленом и синем. Важно подчеркнуть, что на первом сверточном слое используются 32 карты признаков размером 3×3 . Это означает, что каждый нейрон сверточного слоя подключен к квадратному участку изображения размером 3×3 . Второй сверточный слой имеет аналогичную архитектуру, также состоящую из 32 карт признаков с ядром свертки 3×3 . После второго сверточного слоя следует первый слой подвыборки. На этом этапе выполняется уменьшение размерности каждого из признаков. Необходимо отметить, что данный слой также содержит 32 карты признаков, а размер поля выборки составляет 2×2 , что существенно уменьшает размерность данных и способствует более эффективному извлечению ключевых характеристик изображения.

Далее, после первого слоя подвыборки, начинается новый каскад сверточных слоев. Третий и четвертый сверточные слои используют по 64 карты признаков размером 3×3 , что позволяет более детально анализировать входные данные и выделять более сложные признаки.

Второй слой подвыборки, следующий за третьим и четвертым сверточными слоями, также выполняет уменьшение размерности в квадрате 2×2 . После второго слоя подвыборки происходит преобразование данных из двумерного формата в одномерный. Эти одномерные данные затем передаются на полносвязный слой, где и осуществляется окончательная классификация.

Таким образом, вся система работает в гармонии, позволяя эффективно обрабатывать изображения и классифицировать их с высокой степенью точности.

Архитектура разработанной нейросетевой системы представлена на рисунке 5.

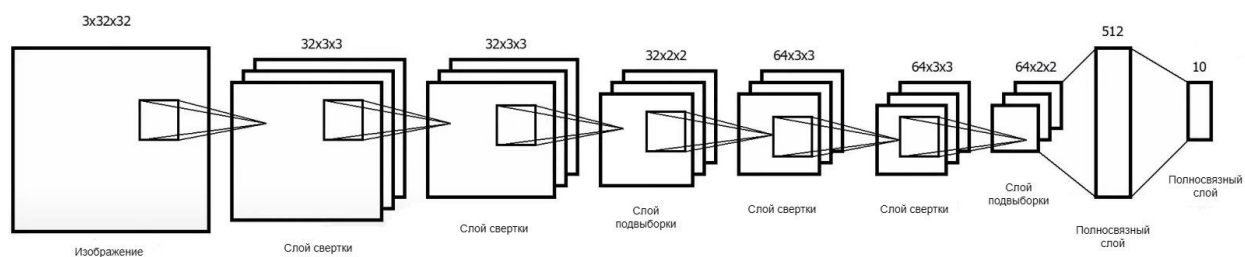


Рисунок 5 – Архитектура нейросетевой системы

Архитектурные особенности разработанной сверточной нейронной сети:

- многоуровневое структурирование: нейросеть состоит из девяти основных элементов, включающих последовательные сверточные и подвыборочные слои, а также полносвязные слои в заключительной части сети. такое разделение на уровни обеспечивает поэтапную обработку входных данных и их преобразование.

- повторяющиеся блоки сверточных слоев и подвыборки: архитектура сети включает два повторяющихся блока, каждый из которых состоит из двух сверточных слоев, за которыми следует слой подвыборки. эти блоки способствуют последовательному выделению и уплотнению признаков изображения.

- разделение на сверточные и полносвязные слои: сверточные слои предназначены для выделения признаков изображения, а полносвязные слои используются для окончательной классификации. это разделение позволяет эффективно сочетать локальное и глобальное обучение признаков.

- сверточные слои с малым размером ядра: каждый сверточный слой содержит карты признаков размером 3x3. это позволяет эффективно выделять локальные признаки на изображении и снижает вычислительную сложность по сравнению с большими ядрами.

- карты признаков на каждом слое: на первом и втором сверточных слоях используется по 32 карты признаков, а на третьем и четвертом слоях — по 64 карты признаков. увеличение числа карт на более поздних слоях позволяет выделять более сложные и высокоуровневые признаки.

- слои подвыборки для уменьшения размерности: каждый слой подвыборки уменьшает размерность данных с помощью выборки 2×2 , что способствует уменьшению объема данных, с которыми работают последующие слои, и увеличению вычислительной эффективности.

- полносвязный слой перед классификацией: после сверточных и подвыборочных слоев данные преобразуются из двумерного формата в одномерный и передаются на полносвязный слой с 512 нейронами. этот слой выполняет интеграцию признаков и подготовку данных к выходной классификации.

- выходной слой для классификации: выходной слой содержит 10 нейронов, каждый из которых представляет вероятность принадлежности изображения к определенному классу. это обеспечивает возможность классификации на 10 категорий.

- входные данные с тремя каналами: на вход сети подаются изображения размером 32×32 пикселя с тремя цветовыми каналами: красным, зеленым и синим. это позволяет обрабатывать цветные изображения и учитывать информацию из всех трех каналов.

Архитектура данной нейросетевой системы, включающая каскад из сверточных и подвыборочных слоев, а также полносвязный классификатор, обеспечивает высокую эффективность и точность в задаче классификации изображений. Такая структура позволяет детально анализировать входные данные и извлекать из них максимально информативные признаки, что является ключевым фактором успешного функционирования нейросетевой модели.

3.2 Реализация и обучение нейросетевой модели

В данном разделе описывается процесс реализации и обучения сверточной нейронной сети (CNN) для классификации изображений одежды из базы данных Fashion MNIST с использованием TensorFlow и Keras.

Для реализации и обучения нейросетевой модели необходимо рассмотреть следующие особенности:

- настройка окружения и выбор библиотек,
- подготовка данных,
- архитектура нейросети,
- компиляция и обучение модели,
- оценка результатов.

Рассмотрим данные особенности подробнее. Для разработки и обучения нейросетевой модели использовались следующие библиотеки:

- tensorflow и keras для реализации нейросети,
- numpy для работы с массивами данных,
- рабочее окружение было настроено для python с установкой необходимых библиотек.

Для обучения и тестирования модели была использована база данных Fashion MNIST, включающая изображения одежды. Рассмотрим основные действия для данной особенности подробнее:

- загрузка данных из базы Fashion MNIST,
- предобработка данных: масштабирование значений пикселей до диапазона от 0 до 1,
- разделение данных на обучающую и тестовую выборки в соотношении 80/20.

Нейросеть была спроектирована для эффективного извлечения признаков из изображений одежды. Рассмотрим основные действия для данной особенности подробнее:

- создание последовательной модели с помощью Keras Sequential API.

- добавление сверточных слоев

Рассмотрим основные действия для компиляции и обучения модели подробнее:

- компиляция модели с оптимизатором SGD, learning rate 0.01, momentum 0.9 и nesterov=True.
- функция потерь categorical_crossentropy для многоклассовой классификации.
- метрика точности (accuracy).
- обучение модели на обучающей выборке в течение 25 эпох с использованием данных для валидации.

Рассмотрим основные действия для оценки результатов:

- оценка точности модели на тестовой выборке.
- вывод результатов точности работы модели на тестовых данных.

Этот раздел подробно описывает каждый этап создания и обучения сверточной нейронной сети для задачи классификации изображений одежды. Полученная модель обладает высокой точностью и способностью эффективно классифицировать изображения из базы данных Fashion MNIST.

В качестве языка разработки был выбран Python – это один из самых популярных языков программирования, который идеально подходит для разработки чат-ботов для Телеграма по причине простоты и чистоты кода: Python очень прост и читаем, что облегчает разработку и отладку. Код на Python обычно занимает меньше строк, чем на многих других языках, что значительно сокращает время разработки,

- библиотеки и фреймворки. Python имеет огромное сообщество разработчиков, которые создали множество библиотек и фреймворков, что заметно упрощает работу. В частности, существуют специализированные библиотеки для работы с API Телеграма, такие как python– telegram– bot.

- мультиплатформенность. Python-кроссплатформенный язык. Это означает, что чат-боты, написанные на Python, могут быть запущены на любой операционной системе, включая Linux, Windows и macOS,

- интеграция с внешними API. Python обладает мощными возможностями для интеграции с внешними API и веб-сервисами,
- асинхронность. Python поддерживает асинхронное программирование, что позволяет улучшить производительность чат-ботов Телеграма, обрабатывающих большое количество запросов.

В качестве среды разработки был выбран PyCharm – это оптимальная среда разработки (IDE) для Python по следующим причинам:

- интеграция со всеми нужными инструментами,
- мощный отладчик,
- поддержка юнит-тестирования,
- удобство работы,
- удобство отслеживания ошибок.

PyCharm предлагает отличную интеграцию со всеми популярными фреймворками и библиотеками для языка программирования Python. Это означает, что разработчики могут легко работать с такими фреймворками, как Django, Flask, и другими. Кроме того, PyCharm прекрасно взаимодействует с системами контроля версий, такими как Git, а также с различными сервисами для непрерывной интеграции и доставки, что делает его незаменимым инструментом для профессиональной разработки и поддержки проектов любого уровня сложности. Одной из ключевых особенностей PyCharm является его мощный и настраиваемый отладчик для Python. Этот инструмент позволяет разработчикам легко находить и исправлять ошибки в коде, что особенно важно при создании сложных приложений, таких как чат-боты. Отладчик поддерживает возможность пошагового выполнения кода, установки точек останова и просмотра значений переменных, что значительно ускоряет процесс разработки и упрощает выявление логических ошибок. Quality assurance (обеспечение качества) играет важную роль в разработке любого программного обеспечения, и чат-боты не являются исключением. PyCharm поддерживает различные фреймворки для модульного тестирования, такие как pytest и unittest. Это позволяет разработчикам писать и запускать

тесты для своего кода прямо в среде разработки, что способствует созданию надежных и стабильных приложений. PyCharm предоставляет множество функций, которые делают работу разработчика более удобной и эффективной. Автодополнение кода помогает быстро писать правильный код, подсветка синтаксиса делает код более читабельным, а автоматическая корректировка ошибок помогает избежать многих распространенных ошибок. Гибкая система настроек позволяет каждому разработчику настроить PyCharm под свои нужды, что делает его универсальным инструментом для различных стилей работы. PyCharm предоставляет возможность проверки кода на ошибки "на лету", что означает, что ошибки в коде могут быть обнаружены и исправлены еще до запуска программы. Интегрированный PEP8-линтер помогает разработчикам следовать стилю кода, рекомендованному для Python, что делает код не только функциональным, но и красивым и понятным для других разработчиков. Это значительно облегчает командную работу и поддержку кода в будущем.

В целом, сочетание Python и PyCharm – это мощный и удобный инструмент для разработки чат-ботов в Телеграме, предлагающий широкий функционал, удобство работы и быстроту разработки.

3.3 Интерфейс ПО

Для взаимодействия пользователя и разработанной системы был реализован интерфейс взаимодействия по средствам бота в мессенджере Telegram с помощью языка Python. Реализация интерфейса системы была реализована по средствам бота в связи с большим количеством преимуществ в сравнении с другими вариантами реализации. Среди всех плюсов можно выделить такие важные качества как простота отправки и получения изображений, встроенные возможности для уведомлений о завершении обработки, безопасность и конфиденциальность данных, а также легкость в интеграции с системами машинного обучения. Пользователи могут легко отправлять изображения в чате с ботом, при этом используя привычный интерфейс мессенджера, которым пользуются более 900 млн пользователей.

Telegram поддерживает различные форматы изображений такие как JPEG, PNG и др., что в свою очередь обеспечивает гибкость при обработке разных типов данных. Асинхронная обработка позволяет обрабатывать несколько запросов одновременно.

Данный механизм позволяет боту работать с несколькими пользователями одновременно, не задерживая процесс работы с ботом. Возможность легкой интеграции с серверными системами позволяет эффективно разделить задачи по приему изображений и их распознаванию, что позволяет сильно повысить производительность и скорость работы всей системы. Использование частных чатов с ботом обеспечивает конфиденциальность отправляемых изображений и получаемых текстовых результатов.

Бот может отправлять уведомления пользователю о том, что обработка изображения завершена и результат готов, что улучшает пользовательский опыт, а в случае длительной обработки, бот может уведомлять пользователя о стадии процесса, поддерживая их информированными. Возможность настройки доступа к боту только для авторизованных пользователей обеспечивает дополнительный уровень безопасности данных.

Интерфейс ПО реализован по средствам бота в Telegram и должен быть интуитивно понятным и доступным. Бот предназначен для выполнения функций распознавания категории одежды по загруженным изображениям. Целевой аудиторией являются пользователи Telegram с потребностью в распознавании категории одежды на фото.

При запуске бот приветствует пользователя приветственным сообщением, предлагая воспользоваться функционалом бота. При выполнении команды /start бот присылает в чат приветственное сообщение: «Привет! Я бот для распознавания категории одежды. Отправьте мне ваше изображение.»

3.4 Тестирование ПО

Целью тестирования является проверка правильности функционирования системы, ее соответствия техническому заданию и

корректности работы. Проверка работоспособности разработанной системы проводилась с использованием разработанных тестовых примеров. Для тестирования корректности работы нейросетевой системы используются входные данные соответствующие вышеуказанным требованиям.

В качестве метода тестирования использовался метод «черного ящика» (поведенческое тестирование), который проверяет работоспособность через пользовательский интерфейс и сравнивает ожидаемые результаты с полученными. Тестирование методом «черного ящика» позволяет находить ошибки, недоступные для метода «белого ящика». Например, если разработчик забыл добавить функциональность, код будет работать правильно, но спецификация останется не выполненной. Тестирование проводится «с позиции пользователя», в следствии чего программы должны быть удобными и понятными для конечного потребителя.

Тестовые примеры в таблицах 3–14 следует использовать для проверки правильности работы программы.

Таблица 3 – Тестовый пример 1

Входные данные	Ожидаемый результат	Полученный результат
Отправка команды /start в чат боту	После ввода и отправки в чат боту команды /start в ответ должно отправиться текст «Привет! Я бот для распознавания категории одежды. Отправьте мне ваше изображение.»	После ввода и отправки в чат боту команды /start в ответ отправился текст «Привет! Я бот для распознавания категории одежды. Отправьте мне ваше изображение.»

Результат выполнения первого тестового примера продемонстрирован на рисунке 6.

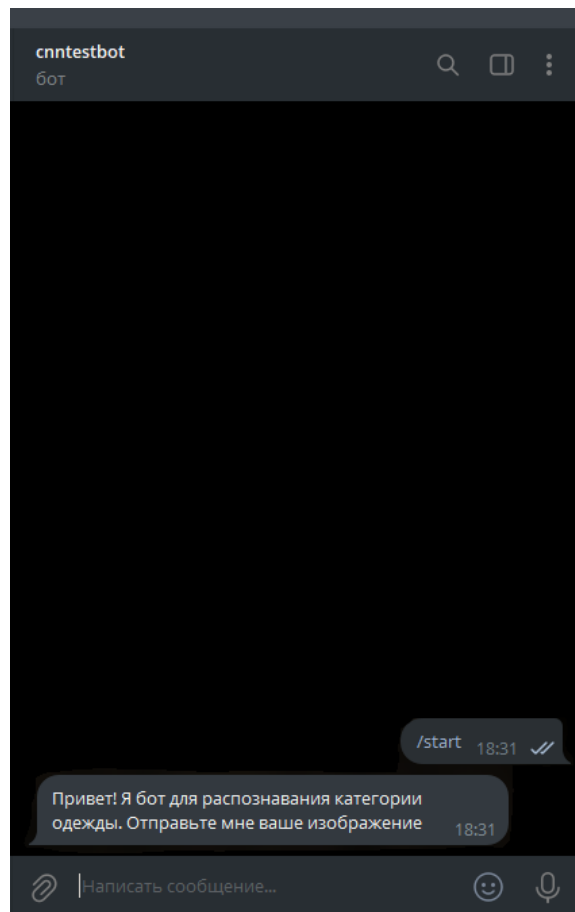


Рисунок 6 – Тестовый пример 1

На рисунке 6 пользователь набирает и отправляет команду /start в чате с Telegram ботом, после чего в ответ пользователю приходит приветственное сообщение с уведомлением пользователя о готовности работы.

Таблица 4 – Тестовый пример 2

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением футболки	После отправки фотографии с изображением футболки система должна вывести результат классификации, а именно текст «футболка»	После отправки фотографии с изображением футболки система выводит результат классификации, а именно текст «футболка»

Результат выполнения второго тестового примера продемонстрирован на рисунке 7.



Рисунок 7 – Результат классификации

Пользователь отправил фотографии футболки и в ответ от бота получил текст «футболка». Этот результат демонстрирует корректность классификации изображения футболки, представленного на рисунке 7, подтверждая тем самым точность работы системы.

Таблица 5 – Тестовый пример 3

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением брюк	После отправки фотографии с изображением брюк система должна вывести результат классификации, а именно текст «брюки»	После отправки фотографии с изображением брюк система выводит результат классификации, а именно текст «брюки»

Результат выполнения третьего тестового примера продемонстрирован на рисунке 8.

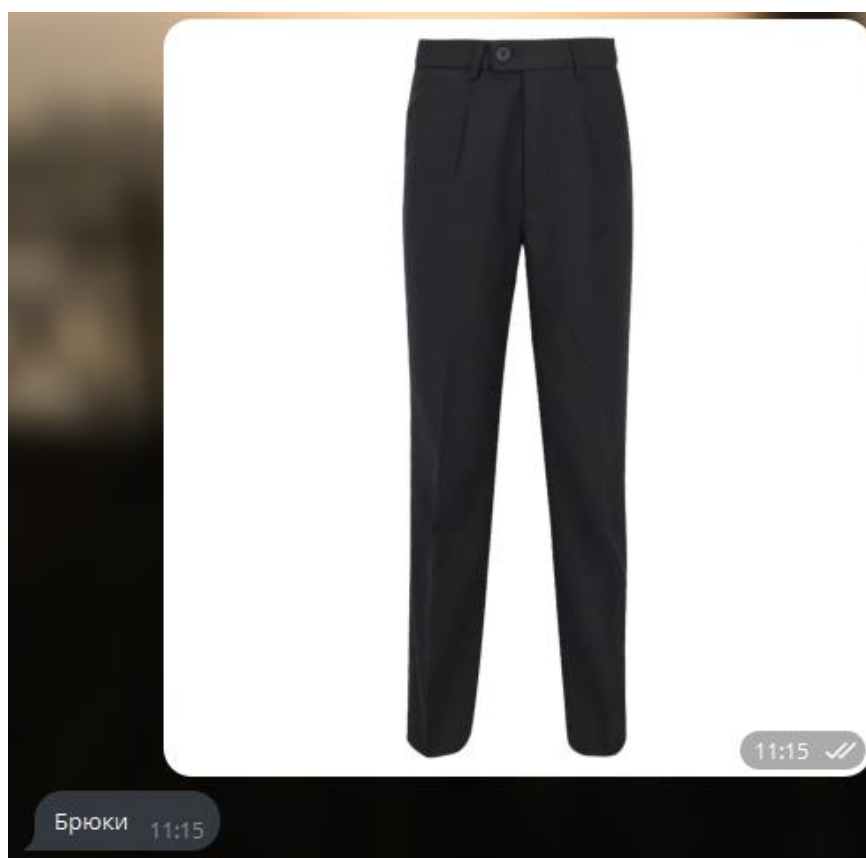


Рисунок 8 – Результат классификации

Пользователь отправил фотографии брюк и в ответ от бота получил текст «брюки». Это говорит нам о корректности классификации выбранного тестового изображения футболки, изображенной на рисунке 8.

Таблица 6 – Тестовый пример 4

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением ботинка	После отправки фотографии с изображением ботинка система должна вывести результат классификации, а именно текст «ботинки»	После отправки фотографии с изображением ботинка система выводит результат классификации, а именно текст «ботинки»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 9.



Рисунок 9 – Результат классификации

Пользователь отправил фотографии ботинка, и в ответ от бота получил текст «ботинки». Этот результат демонстрирует корректность классификации изображения ботинка, представленного на рисунке 9, подтверждая тем самым точность работы системы.

Таблица 7 – Тестовый пример 5

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением рубашки	После отправки фотографии с изображением рубашки система должна вывести результат классификации, а именно текст «рубашка»	После отправки фотографии с изображением рубашки система выводит результат классификации, а именно текст «рубашка»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 10.

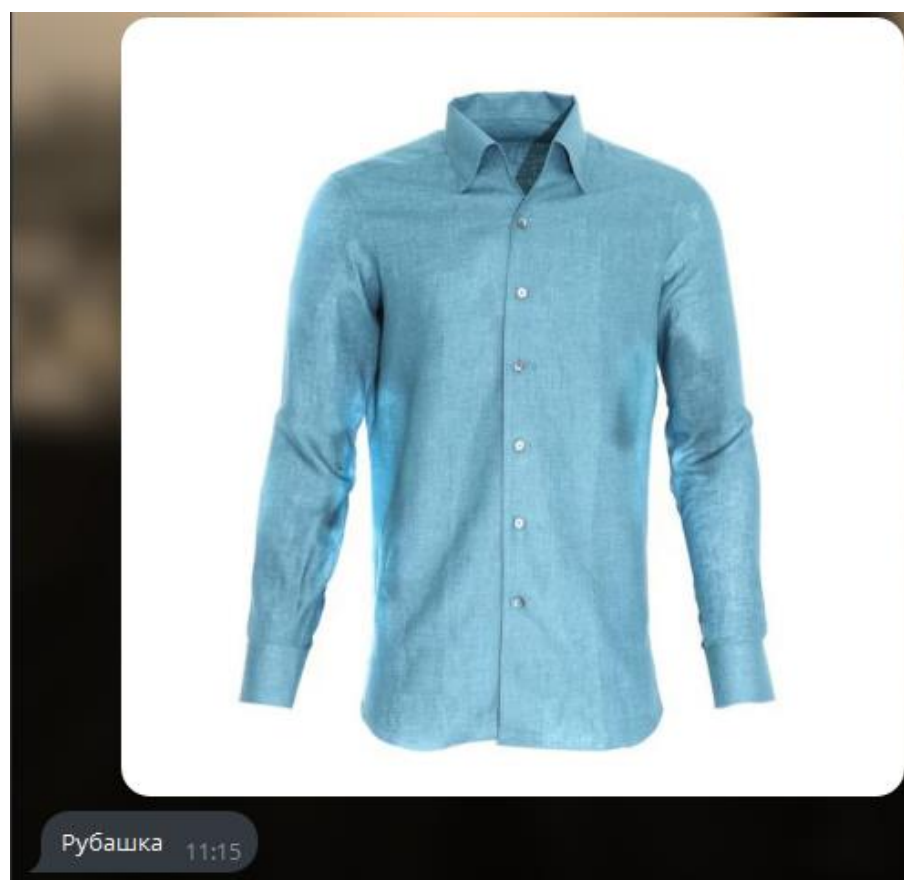


Рисунок 10 – Результат классификации

Пользователь отправил фотографии рубашки и в ответ от бота получил текст «рубашка». Это говорит нам о корректности классификации выбранного тестового изображения футболки, изображенной на рисунке 10.

Таблица 8 – Тестовый пример 6

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением пальто	После отправки фотографии с изображением пальто система должна вывести результат классификации, а именно текст «пальто»	После отправки фотографии с изображением пальто система выводит результат классификации, а именно текст «пальто»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 11.

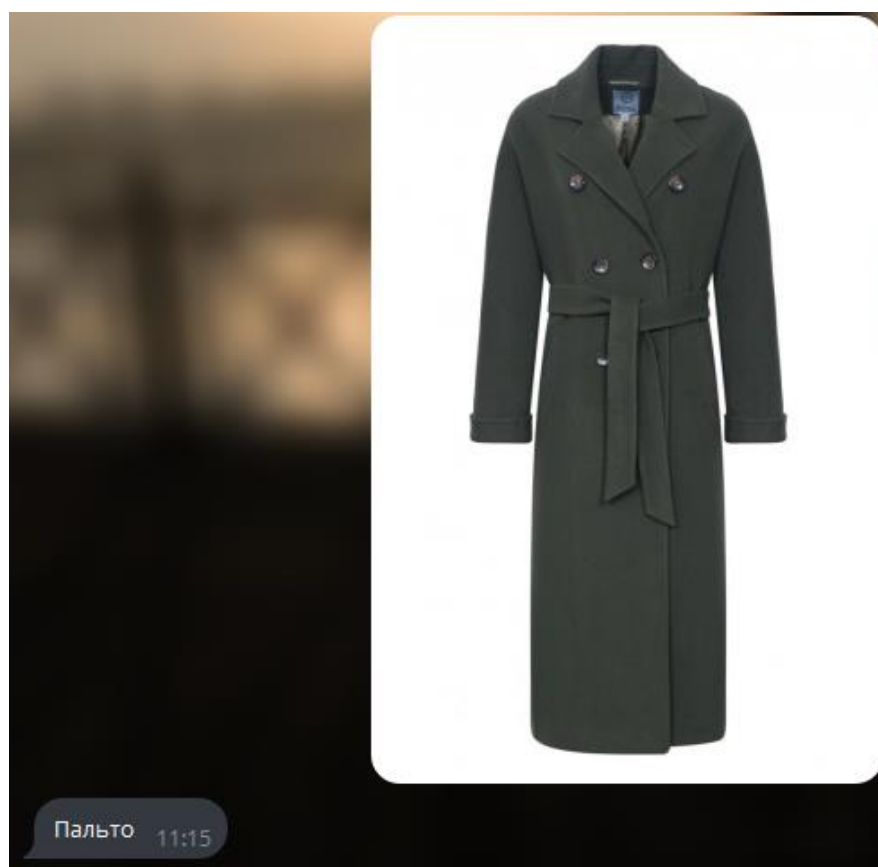


Рисунок 11 – Результат классификации

Пользователь отправил фотографии пальто и в ответ от бота получил текст «пальто». Это подтверждает правильность классификации тестового изображения пальто, показанного на рисунке 11.

Таблица 9 – Тестовый пример 7

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением свитера	После отправки фотографии с изображением свитера система должна вывести результат классификации, а именно текст «свитер»	После отправки фотографии с изображением свитера система выводит результат классификации, а именно текст «свитер»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 12.



Рисунок 12 – Результат классификации

Пользователь отправил фотографии свитера, и в ответ от бота получил текст «свитер». Этот результат демонстрирует корректность классификации изображения свитера, представленного на рисунке 12, подтверждая тем самым точность работы системы.

Таблица 10 – Тестовый пример 8

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением платья	После отправки фотографии с изображением платья система должна вывести результат классификации, а именно текст «платье»	После отправки фотографии с изображением платья система выводит результат классификации, а именно текст «платье»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 13.



Рисунок 13 – Результат классификации

Пользователь отправил фотографии платья, и в ответ от бота получил текст «платье». Этот результат демонстрирует корректность классификации изображения платья, представленного на рисунке 13, подтверждая тем самым точность работы системы.

Таблица 11 – Тестовый пример 9

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением сандали	После отправки фотографии с изображением сандали система должна вывести результат классификации, а именно текст «сандали»	После отправки фотографии с изображением сандали система выводит результат классификации, а именно текст «сандали»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 14.



Рисунок 14 – Результат классификации

Пользователь отправил фотографии сандали, и в ответ от бота получил текст «сандали». Этот результат демонстрирует корректность классификации изображения сандали, представленного на рисунке 14, подтверждая тем самым точность работы системы.

Таблица 12 – Тестовый пример 10

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением кроссовка	После отправки фотографии с изображением кроссовка система должна вывести результат классификации, а именно текст «кроссовки»	После отправки фотографии с изображением кроссовка система выводит результат классификации, а именно текст «кроссовки»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 15.

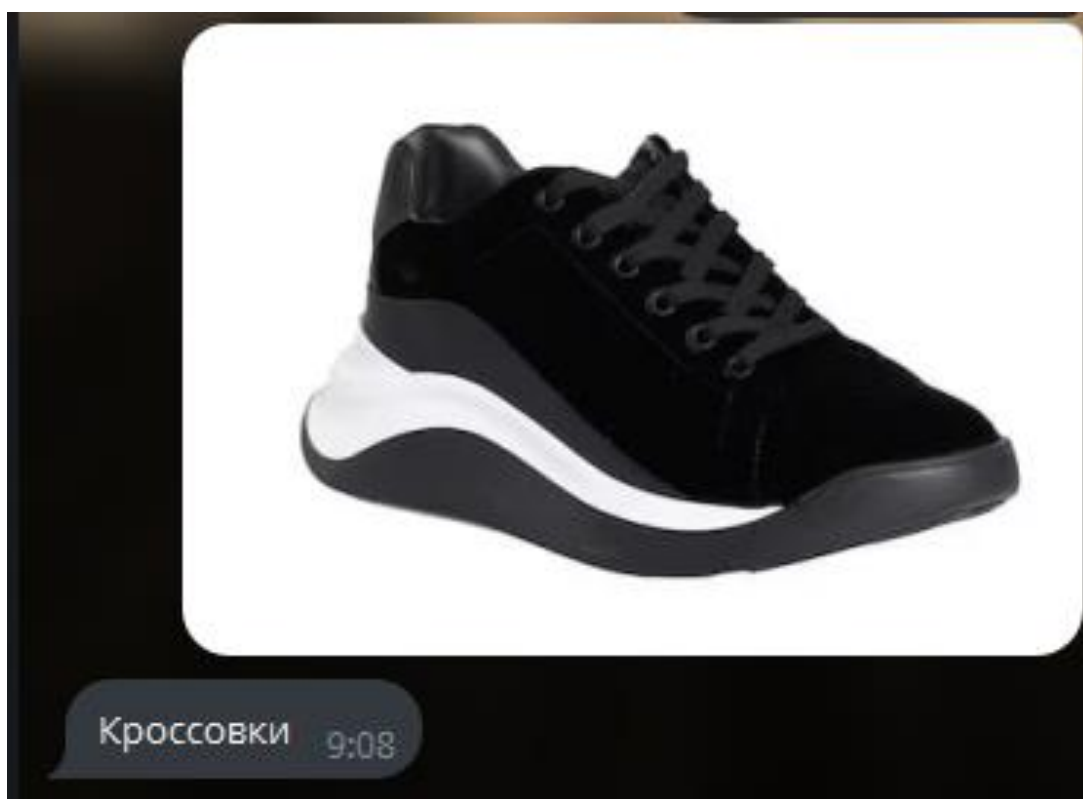


Рисунок 15 – Результат классификации

Пользователь отправил фотографии кроссовка, и в ответ от бота получил текст «кроссовки». Этот результат демонстрирует корректность классификации изображения кроссовка, представленного на рисунке 15, подтверждая тем самым точность работы системы.

Таблица 13 – Тестовый пример 11

Входные данные	Ожидаемый результат	Полученный результат
Фотография с изображением сумки	После отправки фотографии с изображением сумки система должна вывести результат классификации, а именно текст «сумка»	После отправки фотографии с изображением сумки система выводит результат классификации, а именно текст «сумка»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 16.

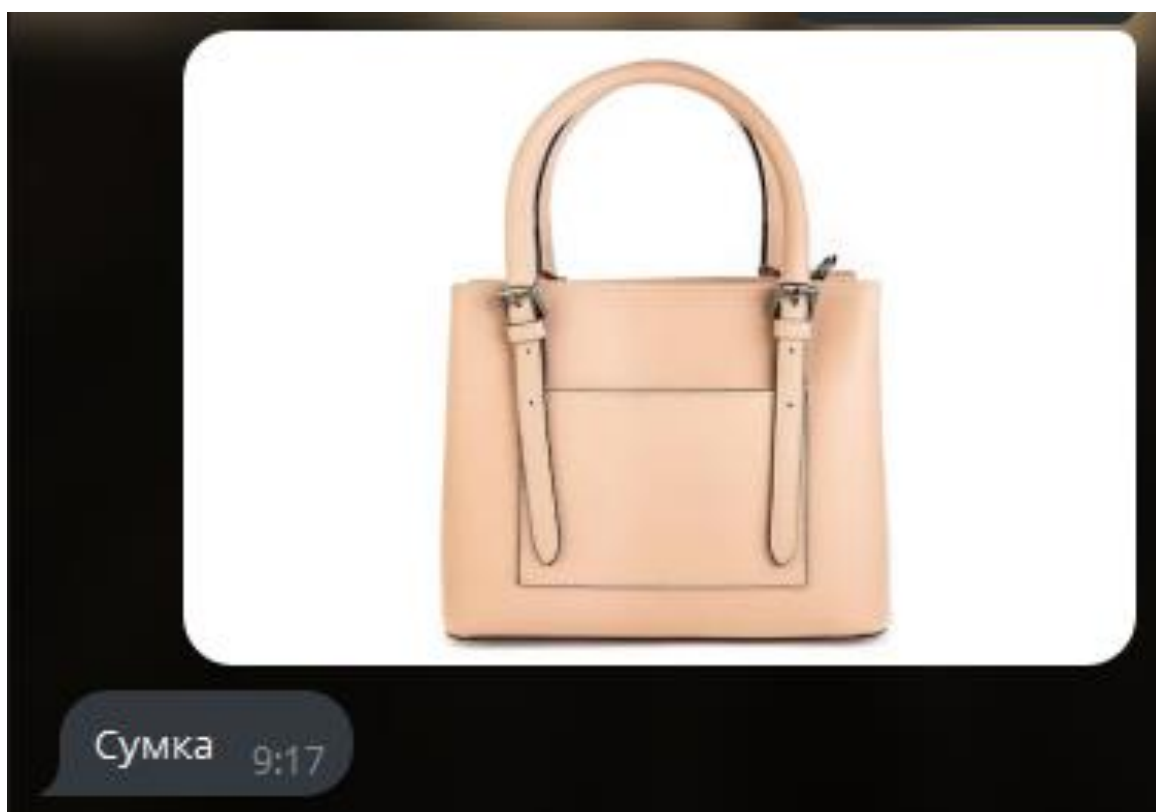


Рисунок 16 – Результат классификации

Пользователь отправил фотографии сумки, и в ответ от бота получил текст «сумка». Этот результат демонстрирует корректность классификации изображения сумки, представленного на рисунке 16, подтверждая тем самым точность работы системы.

Таблица 14 – Тестовый пример 12

Входные данные	Ожидаемый результат	Полученный результат
Фотография без какого-либо элемента одежды	После отправки фотографии без какого-либо элемента одежды система должна вывести результат классификации, а именно текст «пальто»	После отправки фотографии с изображением пальто система выводит результат классификации, а именно текст «пальто»

Результат выполнения четвертого тестового примера продемонстрирован на рисунке 17.

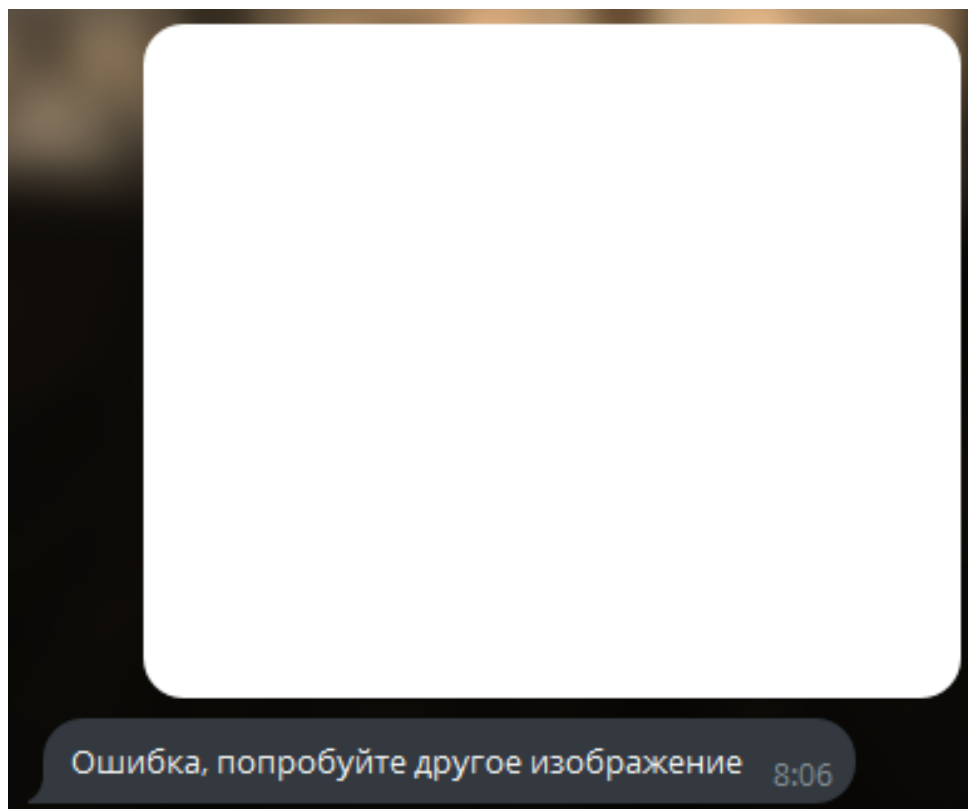


Рисунок 17 – Результат ошибки при классификации

Пользователь отправил фотографии без какого-либо элемента одежды и в ответ от бота получил текст «Ошибка, попробуйте другое изображение». Это подтверждает корректность обработки ошибок при классификации тестового изображения, показанного на рисунке 17.

Таким образом, в результате проведенного тестирования программного обеспечения, разработанного для распознавания образов в нейросетевых системах машинного зрения, удалось убедиться в корректности работы нейросетевой системы.

Была проведена всесторонняя проверка с использованием множества тестовых наборов данных, включающих изображения различных типов одежды, такие как футболки, брюки, платья и куртки. Результаты тестирования показали, что система способна с высокой точностью и надежностью распознавать и классифицировать элементы одежды, предоставляя пользователю точные результаты.

Со всей определенностью можно утверждать, что система корректно классифицирует элементы одежды на классы и информирует пользователя о результатах выполненной классификации.,

ЗАКЛЮЧЕНИЕ

В ходе выполненной работы был произведен анализ предметной области, в рамках которого был произведен анализ различных методов распознавания образов в нейросетевых системах машинного зрения и составлена сравнительная таблица, которая наглядно демонстрирует различия существующих методов и определен наиболее эффективный метод.

Были определены цели и задачи исследования методов распознавания образов в нейросетевых системах машинного зрения, а именно обзор существующих методов, выбор оптимальной модели, разработка и реализация программы. Сформулировано описание и требования к программе к входным данным, к выходным данным и к функционалу.

Была рассмотрена задача автоматического распознавания и категоризации изображений различных предметов одежды с использованием методов нейросетевого распознавания. Для решения этой задачи была выбрана сверточная нейронная сеть (CNN) ввиду ее эффективности, точности и способности извлекать признаки из изображений на разных уровнях абстракции, архитектура выбранной сверточной нейронной сети (CNN) для классификации изображений одежды из базы данных MNIST и работа чат бота для взаимодействия с нейросетевой системой.

Работа подтверждает важность и перспективность применения методов машинного зрения в работе маркетплейсов. Эти методы не только улучшают качество работы специалистов, но и способствуют оптимизации и автоматизации процессов, а также повышению эффективности в широком диапазоне применений.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Кухарев, Георгий Александрович Методы обработки и распознавания изображений лиц в задачах биометрии / Кухарев Георгий Александрович. - М.: Политехника, 2013. - 722 с.
2. Таганов, Александр Иванович Нейросетевые системы искусственного интеллекта в задачах обработки изображений / Таганов Александр Иванович. - М.: Горячая линия - Телеком, 2016. - 722 с.
3. Sankar, K Pal Pattern Recognition From Classical to Modern Approaches / Sankar K Pal, Amita Pal. - Москва: РГГУ, 1983. - 750 с.
4. Scott, E. Umbaugh Computer Imaging: Digital Image Analysis and Processing / Scott E. Umbaugh. - Москва: Наука, 2005. - 659 с.
5. Virginio, Cantoni Artificial Vision: Image Description, Recognition, and Communication (Signal Processing and Its Applications) / Virginio Cantoni, Levaldi Stefano, Vito Roberto. - Москва: Машиностроение, 1981. - 215 с.
6. Гелиг, А. Х. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. Учебное пособие / А.Х. Гелиг, А.С. Матвеев. - М.: Издательство СПбГУ, 2014. - 224 с.
7. Латыпова, Рамиля Нейронные сети / Рамиля Латыпова. - М.: LAP Lambert Academic Publishing, 2012. - 585 с.
8. Редько, В.Г. Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики / В.Г. Редько. - Москва: Машиностроение, 2017. - 224 с.
9. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. - М.: Горячая линия - Телеком, 2013. - 384 с.
10. Латыпова, Рамиля Нейронные сети / Рамиля Латыпова. - М.: LAP Lambert Academic Publishing, 2012. - 585 с.
11. Флах, Петер Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. Учебник / Петер Флах. - М.: ДМК Пресс, 2015. - 400 с.

12. Таганов, Александр Иванович Нейросетевые системы искусственного интеллекта в задачах обработки изображений / Таганов Александр Иванович. - М.: Горячая линия - Телеком, 2016. - 545 с.

13. Финк, Д. Вычислительные машины и человеческий разум: моногр. / Д. Финк. - Москва: РГГУ, 2016. - 296 с.

14. Пестрякова, Н.В. Метод распознавания символов, основанный на полиномиальной регрессии / Н.В. Пестрякова. - Москва: Огни, 2016. - 697 с.

ПРИЛОЖЕНИЕ А

(Обязательное)

Исследование методов распознавания образов в нейросетевых системах

машинного зрения

Листинг программы

Файл «main.py»

```
import numpy

from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Activation,
Dropout
from tensorflow.keras import utils
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers.convolutional import Conv2D,
MaxPooling2D
from tensorflow.keras.utils import np_utils
from tensorflow.keras.optimizers import SGD

import matplotlib.pyplot as plt
%matplotlib inline

# Задаем seed для повторяемости результатов
numpy.random.seed(42)

# Загружаем данные
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
# Размер мини-выборки
batch_size = 32
# Количество классов изображений
nb_classes = 10
# Количество эпох для обучения
nb_epoch = 25
# Размер изображений
img_rows, img_cols = 32, 32
# Количество каналов в изображении: RGB
img_channels = 3
```

```

classes = ['футболка', 'брюки', 'свитер', 'платье', 'пальто',
'туфли', 'рубашка', 'кроссовки', 'сумка', 'ботинки']

# Нормализуем данные
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

# Преобразуем метки в категории
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

# Создаем последовательную модель
model = Sequential()
# Первый сверточный слой
model.add(Conv2D(32, (3, 3), padding='same',
                  input_shape=(32, 32, 3),
                  activation='relu'))
# Второй сверточный слой
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
# Первый слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
# Слой регуляризации Dropout
model.add(Dropout(0.25))

# Третий сверточный слой
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
# Четвертый сверточный слой
model.add(Conv2D(64, (3, 3), activation='relu'))
# Второй слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
# Слой регуляризации Dropout
model.add(Dropout(0.25))
# Слой преобразования данных из 2D представления в плоское
model.add(Flatten())

```

```

# Полносвязный слой для классификации
model.add(Dense(512, activation='relu'))

# Слой регуляризации Dropout
model.add(Dropout(0.5))

# Выходной полносвязный слой
model.add(Dense(nb_classes, activation='softmax'))

# Задаем параметры оптимизации
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

# Обучаем модель
model.fit(X_train, Y_train,
        batch_size=batch_size,
        epochs=nb_epoch,
        validation_split=0.1,
        shuffle=True,
        verbose=2)

# Оцениваем качество обучения модели на тестовых данных
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Точность работы на тестовых данных: %.2f%%" %
      (scores[1]*100))

predictions = model.predict(x_train)

# Меняйте значение n чтобы просмотреть результаты распознавания
других изображений
n = 0
plt.imshow(x_train[n].reshape(32, 32), cmap=plt.cm.binary)
plt.show()

# Печатаем название класса правильного ответа
classes[np.argmax(predictions[n])]

```

ПРИЛОЖЕНИЕ Б

(обязательное)

Исследование методов распознавания образов в нейросетевых системах
машинного зрения

Презентация

