МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Распознавание образов средствами нейронной сети на примере рукописных цифр

Обучающегося 4 курса
очной формы обучения
Будагяна Артема Игоревича

Руководитель выпускной квалификационной работы:
Кандидат технических наук, доцент кафедры информационных технологий и электронного обучения
Карпова Наталья Александровна

Санкт-Петербург
2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время нейронные сети активно внедряются во все сферы деятельности человека, решая при этом самые разнообразные задачи. Список этих задач достаточно обширен: прогноз погоды, редакция снимков и картин, помощь в навигации, чтение речи по артикуляции и многое другое. Одной из самых востребованных задач, в которых нейронные сети широко используются, является распознавание образов. Распознавание образов - это процесс автоматического определения конкретных свойств объектов, таких как форма, размер, цвет и т.д., на основе изображений. Одной из подзадач в этой области является распознавание рукописных цифр.

Распознавание рукописных цифр может иметь широкий спектр применений, таких как распознавание почерка в системах безопасности, обработка изображений в медицинских и биологических исследованиях, распознавание номеров в автоматизированных системах, распознавание лиц и многое другое.

Целью данной работы является исследование возможностей применения нейронных сетей для распознавания рукописных цифр, а также создание компьютерной модели нейронной сети и ее обучение распознаванию рукописных цифр.

Для достижения данной цели были поставлены следующие задачи:

1. Изучить основы создания и функционирования нейронных сетей.
2. Выполнить анализ литературы по созданию и функционированию нейронных сетей.
3. Проанализировать существующие методы распознавания образов и определить их преимущества и недостатки.

4. Выбор структуры нейронной сети для распознавания рукописных цифр.
5. Выбор инструментов и технологий для создания кода нейронной сети.
6. Исследовать существующие наборы данных для обучения нейронной сети и выбрать наиболее подходящий набор.
7. Обучить созданную нейронную сеть распознаванию рукописных цифр.

В данной работе рассматривается распознавание рукописных цифр с помощью нейронной сети. Эта задача была изучена и исследована многими исследователями в прошлом, и на данный момент уже существуют различные методы распознавания рукописных цифр.

Поэтому существует возможность использовать эти решения для создания новых систем, которые будут работать в других сферах. Например, одной из потенциальных возможностей нейронных сетей, способных распознавать рукописные цифры, является сфера образования.

В данной области многие образовательные учреждения используют ручное распознавание цифр, что может приводить к ошибкам и задержкам в обработке данных. Использование нейронных сетей для распознавания цифр может улучшить процесс обработки данных, повысить точность и снизить время, необходимое для выполнения операций.

В рамках работы рассмотрен процесс обучения нейронной сети на примере базы данных MNIST.

База данных MNIST содержит набор изображений рукописных цифр, которые будут использоваться для обучения нейронной сети. В процессе обучения, нейронная сеть находит определенные закономерности в изображениях, которые позволяют ей точно распознавать цифры на новых изображениях.

Несмотря на неоднократное обращение исследователей к решению задачи распознавания рукописных цифр она все еще остается актуальной и

вызывает интерес и необходимость для проведения дополнительных исследований.

В ходе выполнения работы были изучены и использованы публикации современных исследователей Ян Лекун, Боровикова В. П. и Джеффри Хинтон. В работах данных авторов представлены различные методы и алгоритмы обучения нейронных сетей для распознавания образов, в том числе и рукописных цифр. В работе также были проанализированы научные статьи и публикации других авторов, которые рассматривают данную проблему с разных точек зрения.

Объектом исследования данной ВКР является процесс распознавания образов средствами нейронной сети на примере рукописных цифр. Предметом исследования является создание и обучение нейронной сети для распознавания рукописных цифр и возможности ее применения в образовании.

Таким образом, данная работа имеет как теоретическое, так и практическое значение. Теоретически, она содержит обзор методов и технологий создания и обучения нейронных сетей для распознавания образов, а практически, может быть использована для создания более современных систем, способных повысить эффективность и точность обработки данных в образовании.

ГЛАВА 1. ОСНОВЫ НЕЙРОННЫХ СЕТЕЙ И РАСПОЗНАВАНИЯ ОБРАЗОВ

1.1 Понятие нейронных сетей, их структура, технологии создания и возможности их использования

Нейронные сети не всегда получали признание и пользовались популярностью со стороны общества. Только после 2010 года люди начали признавать эффективность работы НС: “Сторонники традиционного машинного обучения перестали высмеивать нейронные сети в 2010 г., когда последние наконец продемонстрировали свою эффективность” [9]. Нейронной сетью называют математическую модель, созданную по принципу функционирования мозга, которая обучается распознавать образы и делать прогнозы на основе определенных параметров. Структура НС пришла в мир программирования прямиком из биологии. Это значит, что архитектура НС повторяет структуру одного из важнейших отделов нервной системы человека. Другими словами, НС это машинная интерпретация человеческого мозга, в котором находятся миллионы нейронов передающих информацию в виде электрических импульсов.

НС состоит из множества элементов, соединенных между собой, которые называются искусственными нейронами или нейронами сети [7]. Искусственные нейроны являются программными модулями, которые также можно назвать узлами. Нейроны - это вычислительная единица, которая принимает информацию, производит над ней простые вычисления и передает ее дальше. Существует три основных типа нейронов:

- Входные;
- Скрытые;
- Выходные.

Когда НС состоит из большого количества нейронов, применяется термин слои. Входные, скрытые и выходные нейроны образуют входной, скрытый и выходной слой соответственно. Входной слой принимает данные из внешнего источника. Затем он обрабатывает эти данные, анализирует или классифицирует их и передает на следующий слой. Скрытые слои в свою очередь получают данные от входного слоя или других скрытых слоев. Они анализируют входные данные предыдущего слоя, обрабатывают и также передают их на следующий слой. Обычно, в НС используется не больше трех скрытых слоев, в случае, когда скрытых слоев больше одного, НС называют глубокой или сетью глубокого обучения. Выходной слой собирает данные предыдущих слоев и возвращает результаты работы сети.

У каждого из нейронов сети есть два основных параметра: входные и выходные данные. Параметры входных нейронов всегда равны, что значит входные данные являются для них выходными. Каждый узел в НС имеет несколько входов и один выход. Каждый вход имеет свой вес. Он указывает на связи одного узла с другим и определяет важность входа для нейрона. Случай, когда вес имеет положительное значение означает, что один узел возбуждает другой, и наоборот, в случае, если вес является отрицательным значением - один узел подавляет другой. Большее значение веса узла оказывает большее влияние на последующие узлы [3].

Разработка и обучение НС - это долгая и объемная работа, в процессе которой используются различные методы и технологии. Некоторыми основными технологиями, используемыми при создании НС являются:

- Python - один из наиболее популярных языков программирования для разработки НС. Он предоставляет широкий набор библиотек и фреймворков для работы с НС, таких, как TensorFlow, PyTorch, Keras и Theano. Эти фреймворки позволяют облегчить процесс создания и обучения НС, предоставляя высокоуровневые API и готовые модели.
- Препроцессинг данных - один из важных этапов создания НС, которая позволяет предварительно обработать данные, поступающие

НС. Этот процесс включает в себя очистку данных от выбросов, масштабирование значений, а также нормализацию и преобразование данных в формат, пригодный для обучения НС.

- Архитектура - определяет структуру и организацию слоев НС и ее нейронов. Существует множество типов архитектур, таких как полносвязные сети, сверточные нейронные сети, рекуррентные нейронные сети и т. д. Выбор архитектуры зависит от конкретной задачи, для которой разрабатывается НС.
- Функция потерь - функция, позволяющая измерять разницу между прогнозами сети и фактическими значениями для максимально точного обучения НС. Для работы этой функции применяются алгоритмы оптимизации, такие как стохастический градиентный спуск, для настройки весов сети и минимизации самой функции потерь. Обучение происходит путем передачи обучающих примеров через сеть и корректировки весов на основе полученных ошибок.
- Методы регуляризации - во время обучения НС может возникнуть ситуация, когда сеть хорошо работает на обучающих данных, но плохо обобщается на новых данных. Для предотвращения переобучения применяются методы регуляризации, такие как добавление слоев снижения переобучения, аугментация данных, обрезка весов и использование регуляризационных функций потерь.
- Оценка производительности - после завершения обучения необходимо оценить производительность НС. Она производится путем тестирования сети на не принадлежащих обучающей выборке данных. Для измерения производительности сети могут быть использованы различные метрики, такие как точность, полнота, F1-мера и ROC-кривая.
- Гиперпараметры - параметры алгоритмов, значения которых устанавливаются перед запуском процесса обучения, например, такие как количество слоев, количество нейронов в слоях, скорость обучения

и размер мини-пакетов. Оптимальная настройка этих гиперпараметров может значительно повлиять на производительность сети. Этого можно достигнуть с помощью опытного подхода, экспериментирования и использования методов оптимизации, таких как перекрестная проверка и случайный поиск.

Нейронные сети имеют широкий спектр применения и могут использоваться в различных областях и сферах:

- Компьютерное зрение: НС являются актуальными для задач распознавания и классификации изображений, обнаружения объектов, сегментации изображений, анализа, их синтеза и многого другого. Их используют, для создания систем видеонаблюдения, автоматического распознавания лиц, медицинской диагностики и обработки изображений в реальном времени.
- Обработка естественного языка: НС применяются для анализа и обработки текстовых данных. Они могут использоваться для задач машинного перевода, определения тональности текста, классификации документов, генераций текстов и других задач, связанных с обработкой языка.
- Рекомендательные системы: НС обладают популярностью в области создания персонализированных рекомендаций. Они могут анализировать предпочтения пользователей, обрабатывать информацию о товарах или контенте и предлагать релевантные рекомендации. Данный метод широко применяется в интернет-магазинах, потоковых сервисах, социальных сетях и различных платформах, которые содержат рекламу.
- Прогнозирование и предсказание: НС могут использоваться для анализа временных рядов и прогнозирования будущих значений. Их применяют в финансовых предсказаниях, прогнозировании погоды, анализе рынка и других областях, где имеется смысл делать прогнозы на основе исторических данных.

- Автономные системы: НС часто используются в создании автономных систем, способных принимать решения и действовать без прямого участия человека. Это могут быть беспилотные автомобили, роботы, автоматизированные системы управления и другие системы, которые обучаются и адаптируются к окружающей среде.
- Медицина и биология: НС находят применение в медицине и биологии в задачах, таких как диагностика и прогнозирование заболеваний, анализов медицинских изображений (например, рентгеновских снимков или снимков МРТ), моделирования и прогнозирования биологических процессов, анализов генетических данных и многих других. НС позволяют обрабатывать большие объемы данных и выявлять сложные взаимосвязи, что помогает в современных медицинских исследованиях и практиках.
- Финансы и банковское дело: НС применяют для прогнозирования финансовых рынков, оценки рисков, определения кредитоспособности клиентов, обнаружения мошенничества и других задач, связанных с финансовой аналитикой и банковским делом. Они способны анализировать большие объемы данных и выявлять сложные закономерности, что помогает в принятии более точных решений.
- Промышленность и производство: НС применяются в промышленности и производстве для оптимизации процессов, контроля качества, прогнозирования отказов оборудования, оптимизации энергопотребления и т.д. Они способны адаптироваться к изменяющимся условиям и находить оптимальные решения в реальном времени, что повышает эффективность и надежность производственных процессов.

1.2 Основы распознавания образов, алгоритмы и технологии их реализации

Распознавание образов — процесс выявления и классификации закономерностей в данных. Он стал важной областью компьютерных наук и искусственного интеллекта, поскольку имеет множество практических применений, таких как распознавание изображений, распознавание речи и обработка естественного языка [8].

Изначально, понятие распознавания образов не было привязано к компьютерным наукам, так как это понятие появилось задолго до возникновения компьютерных систем и технологий. Стоит отметить, что распознавание образов всегда являлось и является частью деятельности мозга человека. Первые методы распознавания начали разрабатываться для электронных аналоговых систем и рассматривались в рамках теории обработки сигналов.

Алгоритмы распознавания образов — это вычислительные процедуры, предназначенные для анализа и выявления закономерностей в данных [12].

Алгоритмы распознавания образов можно разделить на несколько категорий в зависимости от подхода и задачи:

1. Методы основанные на признаках (Feature-based methods): эти алгоритмы анализируют изображение, выделяя характерные признаки, такие как границы, углы, текстуры или цвета. После чего эти признаки сравниваются с шаблонами или моделями, чтобы определить, к какому классу или категории принадлежит объект.
2. Методы основанные на шаблонах (Template-based methods): в этом подходе предварительно создаются шаблоны объектов, и только затем изображение сравнивается с этими шаблонами. Часто используется сопоставление шаблона с помощью корреляции или методов сравнения.

3. Методы основанные на статистике (Statistical methods): эти алгоритмы используют статистические модели и методы для анализа и классификации объектов. Они могут включать в себя методы машинного обучения, такие как методы байесовской классификации, методы опорных векторов или НС.
4. Методы основанные на нейронных сетях (Neural network-based methods): с использованием глубоких нейронных сетей, таких как сверточные нейронные сети (CNN), можно достичь высокой точности в распознавании образов. Эти методы требуют большого количества размеченных данных для обучения модели, но они способны автоматически извлекать признаки из изображений и достигать хороших результатов в сложных задачах распознавания [18].

В реализации распознавания образов используются различные технологии, включая:

1. Извлечение признаков (Feature extraction): процесс предварительной обработки изображений, который позволяет выделить характерные признаки для дальнейшего анализа и классификации. Он может включать фильтрацию, масштабирование, преобразования цветового пространства и другие методы.
2. Классификация (Classification): этап, включающий в себя обучение моделей для классификации объектов на основе извлеченных признаков. Методы классификации могут варьироваться от простых алгоритмов, таких как метод ближайших соседей, до более сложных, таких как случайные леса, градиентный бустинг или нейронные сети.
3. Обучение с учителем и без учителя (Supervised and unsupervised learning): для обучения моделей распознавания образов используются различные методы машинного обучения. В случае обучения с учителем, модель обучается на размеченных данных, где каждый объект имеет известную метку класса или маркировку. В случае

обучения без учителя, модель ищет скрытые структуры или паттерны в данных без заранее известных меток классов [2].

4. Оценка и оптимизация (Evaluation and optimization): для оценки качества алгоритмов распознавания образов используются различные метрики, такие как точность, полнота, F-мера, ROC-кривая и другие. Для улучшения результатов могут применяться методы оптимизации, которые включают в себя настройку гиперпараметров модели, аугментацию данных и ансамблирование моделей.

Также существуют другие алгоритмы и технологии, которые используются для распознавания образов, такие, как деревья решений, машины опорных векторов и глубокое обучение. Деревья решений — это древовидные структуры, которые можно использовать для классификации данных, задавая ряд вопросов, на которые можно ответить «да» или «нет», в то время, как машины опорных векторов — это модели, которые можно использовать для разделения данных на разные классы на основе их характеристик.

Кроме алгоритмов распознавания образов таких, как обучение с учителем и обучение без учителя существует алгоритм обучения с подкреплением. Алгоритмы обучения с подкреплением обучаются с помощью системы вознаграждений и наказаний, где алгоритм учится максимизировать вознаграждения и минимизировать наказания.

Алгоритм распознавания образов под названием "обучение с подкреплением" (reinforcement learning) - это метод машинного обучения, в котором агент обучается взаимодействуя с окружающей средой, получая обратную связь в виде награды или штрафа. Этот подход в основном используется для задач последовательного принятия решений, где агент должен определить оптимальное действие в каждом состоянии, чтобы максимизировать накопленную сумму наград.

Основные компоненты алгоритма обучения с подкреплением:

- Агент (Agent) - алгоритм или модель, которая принимает решения и взаимодействует с окружающей средой. Агент наблюдает текущее состояние среды, выбирает действие и получает награду в зависимости от своих действий.
- Среда (Environment) - контекст, с которым агент взаимодействует. Среда может быть физической, виртуальной или математической. Она определяет динамику состояний и наград, которые получает агент в ответ на его действия.
- Состояние (State) - состояние среды представляет текущую информацию, необходимую агенту для принятия решения. В зависимости от задачи, состояние может быть полностью наблюдаемым или частично наблюдаемым.
- Действие (Action) - действие является выбором, который агент может совершить в определенном состоянии. Действия могут быть дискретными или непрерывными, в зависимости от задачи.
- Награда (Reward) - награда представляет собой числовую оценку, которую агент получает от среды в ответ на совершенное им действие. Цель агента - максимизировать сумму наград на протяжении всего процесса обучения.

Существуют основные этапы алгоритма обучения с подкреплением:

1. Наблюдение состояния: агент наблюдает текущее состояние среды.
2. Выбор действия: агент выбирает действие на основе своей стратегии, которая может быть определена как функция состояния или функция состояния и времени.
3. Взаимодействие со средой: агент выполняет выбранное действие и переходит в новое состояние.
4. Обновление стратегии: агент обновляет свою стратегию на основе полученной награды и выбранного действия.

5. Повторение процесса: агент повторяет процесс наблюдения, выбора действия, взаимодействия с средой и обновления стратегии на протяжении множества эпизодов или шагов.

Обучение с подкреплением широко используется в задачах игр, робототехнике, автономной навигации и других областях, где требуется принятие последовательных решений в неопределенных средах.

Существует много подходов и методов в этой области, и они продолжают развиваться с развитием компьютерного зрения и искусственного интеллекта.

Распознавание образов - область, требующая глубокого понимания алгоритмов и технологий. Использование НС, деревьев решений, машин опорных векторов, глубокого обучения и других алгоритмов могут быть по своему эффективны для распознавания образов в зависимости от конкретных задач и анализируемых данных.

1.3 Распознавание образов с помощью нейронных сетей

Распознавание образов с помощью НС - это метод, который использует глубокие нейронные сети для анализа и классификации визуальной информации, такой как изображения или видео. НС в этом контексте извлекают признаки из входных данных и принимают решения на основе этих признаков.

Основными компонентами распознавания образов с помощью нейронных сетей являются:

1. Предварительное обучение (Pretraining): в процессе обучения НС для распознавания образов обычно используется предварительное обучение на большом наборе размеченных данных. Модели, такие как VGG, ResNet или Inception, предварительно обучаются на больших наборах данных, таких как ImageNet, где извлекаются общие признаки

изображений. Затем эти предварительно обученные модели могут быть дообучены или настроены на более специфических наборах данных или задачах распознавания образов.

2. Обратное распространение ошибки (Backpropagation): обучение НС для распознавания образов осуществляется путем минимизации функции потерь с использованием алгоритма обратного распространения ошибки. В процессе обратного распространения ошибки градиенты ошибки вычисляются вдоль сети, и веса нейронов обновляются с целью минимизации ошибки.
3. Данные для обучения: для обучения НС для распознавания образов требуются большие объемы размеченных данных. Обычно используются наборы данных, такие как MNIST, CIFAR-10, ImageNet и другие, содержащие изображения с различными классами. Наборы данных должны быть разнообразными и представлять классы, которые требуется распознать.
4. Техники регуляризации: для предотвращения переобучения и улучшения обобщающей способности моделей распознавания образов могут применяться различные техники регуляризации, такие как отсев (dropout), регуляризация весов (weight regularization), аугментация данных (data augmentation) и другие.
5. Выбор функции активации: нейроны в НС распознавания образов обычно имеют функции активации, такие как ReLU (Rectified Linear Unit) или сигмоидальные функции. Функции активации вносят нелинейность в сеть и дают модели возможность аппроксимировать сложные зависимости в данных.

Алгоритмы распознавания образов с помощью НС представляют собой специальные процедуры, которые позволяют ей обучаться и делать прогнозы на основе входных образов или шаблонов. Некоторые из наиболее популярных алгоритмов и технологий, используемых для реализации распознавания образов с помощью НС приведены ниже:

- Нейронные сети прямого распространения (Feedforward): самый простой тип НС, где информация передается от входного слоя через скрытые слои к выходному слою без обратных связей. Алгоритм обработки данных в такой сети включает проход вперед, где данные передаются от одного слоя к другому, и выходные значения вычисляются на основе весов и функций активации нейронов.
- Сверточные нейронные сети (Convolutional Neural Networks, CNN): этот тип НС обычно используется для обработки и анализа изображений. Он содержит слои свертки, которые применяют фильтры для выделения определенных признаков изображения. CNN имеют свойства инвариантности к сдвигу и локальности, что позволяет им успешно работать с изображениями различных размеров и распознавать паттерны в них [1].
- Рекуррентные нейронные сети (Recurrent Neural Networks, RNN): этот тип НС имеет обратные связи, что позволяет использовать информацию о предыдущих входах для принятия решений. RNN широко применяются в задачах обработки последовательностей данных, таких как распознавание рукописного текста, распознавание речи и машинный перевод.
- Глубокое обучение (Deep Learning): подход к обучению НС, который основывается на использовании большого количества слоев и параметров. Глубокое обучение позволяет НС автоматически извлекать иерархические представления данных и строить более сложные модели распознавания образов. Глубокое обучение стало основой для многих современных технологий распознавания образов, таких как распознавание лиц, автоматическая классификация изображений и голосовое управление.
- Библиотеки и фреймворки: для реализации алгоритмов распознавания образов с помощью НС используются различные библиотеки и фреймворки, которые предоставляют готовые инструменты и функции

для создания, обучения и применения нейронных сетей. Некоторые из популярных фреймворков включают TensorFlow, PyTorch, Keras и Caffe.

ГЛАВА 2. РАЗРАБОТКА НЕЙРОННОЙ СЕТИ

2.1 Обоснование выбора инструментов для разработки

Python

Язык программирования Python является одним из популярных высокоуровневых языков программирования. Его зачастую используют в области разработки НС и машинного обучения. Язык дает возможность разрабатывать сложные алгоритмы за относительно короткое время. С помощью Python можно разрабатывать как простейшие нейронные сети, так и сложные с различными архитектурами [11]. Python обладает множеством встроенных библиотек, что позволяет использовать все его возможности для разработки проектов. Одним из преимуществ этого языка можно выделить его простоту, выразительность и лаконичность [10].

Фреймворк TenserFlow

TenserFlow - одна из самых популярных библиотек для задач машинного обучения. Она имеет открытый исходный код и предоставляет широкие возможности для создания и обучения нейронных сетей с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия [21]. Библиотека обладает большой гибкостью и мощью, что позволяет создавать разные типы НС с различными архитектурами. Помимо этого, фреймворк обеспечивает высокую производительность и масштабируемость, что позволяет обрабатывать как маленькие объемы данных, так и большие. Благодаря поддержке различных платформ, таких как CPU, GPU и TPU, TenserFlow способен использовать мощности параллельных вычислений и ускорять процесс обучения [13].

Библиотека MNIST

MNIST (Modified National Institute of Standards and Technology database) - широко используемый набор данных для обучения и тестирования моделей машинного обучения, в особенности моделей распознавания образов цифр. В набор данных входят изображения с рукописными цифрами от 0 до 9 и метками классов на соответствующих изображениях. В наборе данных MNIST содержится 70000 изображений. 60000 из них являются изображениями для обучения, а 10000 изображений выступают в качестве тестового набора. Все изображения в базе данных заранее обработаны и представлены в серых оттенках для более удобного и качественного распознавания. Все цифры в наборе написаны в различных стилях разными людьми, чтобы обеспечить разнообразие данных надежное и эффективное обучение сети. Благодаря этому, после обучения НС становится более устойчивой к вариациям входных данных. Стоит отметить, что набор данных MNIST поддерживается вышеупомянутым фреймворком TenserFlow, что значительно упрощает и ускоряет разработку нейронной сети.

Тулкит Tkinter

Tkinter (Tk interface) - исходя из названия сразу становится ясно, что данный инструмент представляет собой графический интерфейс. Этот модуль работает на платформе Python и включает в себя различные графические компоненты, такие как, кнопки, текстовые поля, скроллеры и многое другое, все эти компоненты можно по другому назвать виджетами. Tkinter по умолчанию является включенным в стандартную библиотеку Python в виде отдельного модуля. Главным преимуществом тулкита является его кроссплатформенность, он работает одинаково как на Windows, так и на Mac OS или Linux. Этот пакет поможет просто и быстро создать приложение для использования ранее разработанной нейронной сети, с которым сможет взаимодействовать конечный пользователь. Tkinter обладает интуитивно понятной структурой, что значительно упрощает его использование.

2.2 Описание алгоритма и технологии разработки

Для создания нейронной сети будет разработана сверточная нейронная сеть. Этот тип сети хорошо подходит для реализации задачи распознавания цифр. CNN специально разработаны для обработки и анализа изображений: они содержат сверточные слои и слои субдискретизации, которые способствуют извлечению важных признаков с изображений и уменьшению размерности данных.

Первым шагом разработки является загрузка данных: необходимо загрузить набор данных, содержащих изображения цифр и соответствующие им метки. Для этого следует импортировать библиотеки TenserFlow и MNIST. После этого стоит провести предобработку данных, а именно нормализовать значения пикселей, масштабируя их от 0 до 1.

Следующим шагом является создание архитектуры CNN. Сначала инициализируется пустая модель, после чего добавляется новый сверточный слой с указанием числа фильтров, размера филтьров и функцией активации, в данном случае функцию ReLU. Также стоит добавить слой активации softmax для получения вероятностей классов.

После создания моделей, производится их компиляция: определяется функция потерь, выбирается оптимизатор и в итоге компилируется модель. Когда модель готова, можно приступать к ее обучению и оценке. В каждую эпоху модель будет обучаться на пакетах данных, обновлять веса и показывать прогресс обучения. При оценке модели используется тестовый набор данных, должны быть выведены значения выбранной метрики оценки, в данном случае точности.

Последними шагами являются настройка и оптимизация сети и тестирование модели. При необходимости нужно произвести настройку и оптимизацию модели, изменяя гиперпараметры, такие как количество слоев, размерность фильтров или количество эпох обучения. Для тестирования

модели используются неразмеченные данные из набора и получение предсказаний модели.

Чтобы использовать нейронную сеть нужно создать приложение с интуитивно понятным интерфейсом простому пользователю. Для быстрой и простой реализации алгоритма его разработки понадобится библиотека Tkinter. Первым шагом будет являться создание окна приложения и элементов управления, таких как кнопка загрузки фотографий цифры и метка для отображения результатов распознавания. При нажатии кнопки будет вызываться функция загрузки изображения и его преобразования в подходящий для работы НС формат.

После этого в приложение загружается ранее обученная модель распознавания цифр с использованием TenserFlow. Модель будет использована для предсказания распознавания цифры на обработанном изображении. Следующим шагом является отображение предсказанного результата распознавания на метке.

Одним из важных этапов разработки приложения является его тестирование и отладка, при некорректной работе приложения стоит внести исправления в код для его исправления и улучшения. Последним шагом станет развертывание приложения: создание исполняемого файла или пакета установки на других устройством является заключительным этапом разработки приложения для нейронной сети.

2.3 Обучение нейронной сети распознаванию рукописных цифр

Первым шагом разработки нейронной сети является импорт всех необходим библиотек и пакетов для работы с изображениями и обучением нейронной сети. Как уже упоминалось выше, нам понадобятся такие библиотеки и пакеты, как TensorFlow и MNIST. Импорт библиотек можно увидеть на рисунке 1.

```
import tensorflow as tf
import numpy as np
from PIL import Image
import time
from tensorflow.keras.models import load_model
import cv2
import matplotlib.pyplot as plt
```

Рисунок 1 - Импорт библиотек

Кроме них также будут использоваться библиотеки OpenCV для загрузки изображения и отображения результатов работы сети во время ее обучения и PILLOW для работы с изображениями.

После импортирования библиотек начинается загрузка данных MNIST, включая наборы тренировочных и тестовых изображений и соответствующих им меток. Данный этап отображен на рисунке 2.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Рисунок 2 - Загрузка данных

Чтобы нейронная сеть могла работать с изображениями, сначала их нужно нормализовать и изменить размер на 28x28 пикселей. Этот процесс отображен на рисунке 3.

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1).astype('float32') / 255.0
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Рисунок 3 - Обработка изображений

Эти строки выполняют предобработку данных. Они изменяют форму изображений, нормализуют значения пикселей и преобразуют метки в категориальный формат.

После загрузки и обработки всех необходимых данных можно приступать к созданию модели сверточной нейронной сети. Процесс ее создания можно увидеть на рисунке 4.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Рисунок 4 - Создание модели нейронной сети

Эти строки создают модель сверточной нейронной сети с использованием Sequential API из Keras. Они добавляют слои свертки, пулинга, плоского слоя и полносвязных слоев.

Следующим шагом разработки является компиляция и обучение модели. Данный процесс изображен на рисунке 5.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10, batch_size=128, validation_data=(x_test, y_test))
```

Рисунок 5 - Обучение модели

Эти строки компилируют модель, указывая оптимизатор, функцию потерь и метрики. Затем модель обучается на тренировочных данных в течение заданного числа эпох.

Когда модель обучена, ее можно сохранить в файле модели в бинарном формате с расширением “.h5”. При сохранение модели в файл используется метод “save()” из библиотеки Keras, как показано на рисунке 6.

```
model.save("model.h5")
print("Модель сохранена, как Model.h5")
time.sleep(10)
```

Рисунок 6 - Сохранение модели

После разработки модели нейронной сети стоит запустить программу и проверить насколько хороша она научилась распознавать цифры на изображениях. Обучение модели с количеством эпох равным десяти занимает около 2–3 минут, при увеличении количества эпох, увеличится и время, затрачиваемое моделью на обучение. Слишком большое количество эпох

может негативно повлиять на нейронную сеть и начать процесс переобучения.

```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\python3.10.exe
Epoch 1/10
469/469 [==============================] - 18s 36ms/step - loss: 0.2407 - accuracy: 0.9290 - val_loss: 0.0578 - val_accuracy: 0.9818
Epoch 2/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0623 - accuracy: 0.9809 - val_loss: 0.0509 - val_accuracy: 0.9832
Epoch 3/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0454 - accuracy: 0.9856 - val_loss: 0.0351 - val_accuracy: 0.9881
Epoch 4/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0355 - accuracy: 0.9887 - val_loss: 0.0366 - val_accuracy: 0.9881
Epoch 5/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0294 - accuracy: 0.9906 - val_loss: 0.0298 - val_accuracy: 0.9895
Epoch 6/10
469/469 [==============================] - 17s 35ms/step - loss: 0.0234 - accuracy: 0.9927 - val_loss: 0.0301 - val_accuracy: 0.9902
Epoch 7/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0224 - accuracy: 0.9928 - val_loss: 0.0280 - val_accuracy: 0.9917
Epoch 8/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0174 - accuracy: 0.9944 - val_loss: 0.0289 - val_accuracy: 0.9913
Epoch 9/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0169 - accuracy: 0.9943 - val_loss: 0.0335 - val_accuracy: 0.9889
Epoch 10/10
469/469 [==============================] - 17s 36ms/step - loss: 0.0128 - accuracy: 0.9959 - val_loss: 0.0353 - val_accuracy: 0.9885
Модель сохранена, как Model.h5
```

Рисунок 7 - Процесс обучения нейронной сети

Исходя из рисунка 7 видно, что с каждой новой эпохой, точность обучения увеличивается, а потери уменьшаются.

2.4 Разработка приложения для использования нейронной сети

Для разработки приложения понадобится тулкит Tkinter и остальные необходимые библиотеки. Загрузка библиотек показана на рисунке 8.

```
1  import cv2
2  import tensorflow as tf
3  from tkinter import Tk, Label, Button, filedialog
4  import numpy as np
5  from PIL import Image, ImageTk
```

Рисунок 8 - Импорт библиотек

Библиотека PIL понадобится в ходе разработки для работы с изображениями.

Для функционирования приложения необходимо загрузить в файл модель ранее разработанной нейронной сети (рисунок 9).

```
8  model = tf.keras.models.load_model('model.h5')
```

Рисунок 9 - Импорт модели

```
def select_image():
    root = Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename()

    image = cv2.imread(file_path)

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)

    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    recognized_digits = []
```

Рисунок 10 - Загрузка и подготовка изображения

Функция select_image с рисунка 10 будет содержать в себе все необходимые процессы для преобразования изображений, создание диалогового окна выбора изображения в программе и процесс распознавания самих цифр. С помощью функции OpenCV “cv2.imread” происходит загрузка изображения в программу, после чего с помощью функции “cv2.cvtColor” изображение преобразуется в оттенки серого. Но для более точной работы сети есть смысл применить алгоритм бинаризации и получить черно-белое изображение без лишних пикселей серого цвета. Так как нейронная сеть разрабатывается не просто для определения одной цифры с картинки, а для работы в сфере образования, то она должна уметь различать сразу несколько цифр, находящиеся на одной картинке. Для их разделения используется алгоритм с контурами. С помощью функции “cv2.findContours” программа находит контуры каждой цифры и определяет какое количество символов нужно будет распознать для выдачи результата. Для корректного отображения результата в программе заранее создадим переменную, в которую будем вносить все распознанные цифры в будущем.

```
    for i, contour in enumerate(contours):

        (x, y, w, h) = cv2.boundingRect(contour)

        digit = binary[y:y+h, x:x+w]

        height, width = digit.shape[:2]
        size = max(height, width)

        square_image = np.zeros((size, size), dtype=np.uint8)

        offset_x = (size - width) // 2
        offset_y = (size - height) // 2

        square_image[offset_y:offset_y+height, offset_x:offset_x+width] = digit

        resized_image = cv2.resize(square_image, (28, 28), interpolation=cv2.INTER_AREA)

        digit_image = (1 - resized_image / 255.0)

        digit_image = digit_image / 255.0

        digit_image = cv2.normalize(digit_image, None, 0, 255, cv2.NORM_MINMAX, dtype=cv2.CV_8U)

        digit = digit_image.reshape((28, 28, 1))

        prediction = model.predict(np.array([digit]))
        recognized_digit = np.argmax(prediction)

        recognized_digits.insert(0, recognized_digit)
```

Рисунок 11 - Преобразование и предсказывание цифр

Следующим шагом является преобразованием каждой цифры к верному формату (рисунок 11). Как только программа поняла с каким количеством отдельных изображений ей предстоит работать, она начинает процесс обработки. Для каждой цифры получается ограничивающий ее прямоугольник для текущего контура, после чего она извлекается из ограничивающего прямоугольника. Чтобы загрузить изображение в нейронную сеть, оно должно быть размерами 28x28 пикселей. При сжатии или растягивании изображения может произойти деформирования символа, поэтому сначала исходное изображение помещается на новый слой белого цвета, чтобы получился квадрат. После этого картинка сжимается до нужного размера и начинается процесс нормализации пикселей до диапазона от 0 до 1. Последним этапом перед подачи цифры в модель является изменение ее размерности. Когда все подготовки закончены, начинается

распознавание цифры с помощью ранее обученной модели. После получения результата распознавания цифра записывает в созданную перед началом цикла переменную “recognized_digits”. Цикл будет работать до тех пор, пока не пройдет по всем найденным контурам на изображении, то есть у программы нет ограничения в количестве цифр для распознавания с одной картинки.

```
        digits_text = ", ".join(str(digit) for digit in recognized_digits)
        result_label.config(text="Распознанные цифры: " + digits_text)

        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(image)
        image = ImageTk.PhotoImage(image)
        image_label.config(image=image)
        image_label.image = image

root = Tk()
root.title("Разделение и распознавание цифр")
root.geometry("400x300")

label = Label(root, text="Выберите изображение с цифрами")
label.pack(pady=10)

button = Button(root, text="Выбрать изображение", command=select_image)
button.pack()

image_label = Label(root)
image_label.pack()

result_label = Label(root, text="")
result_label.pack(pady=10)

root.mainloop()
```

Рисунок 12 - Создание графического интерфейса

Когда цифры на изображении кончатся приложение выдаст результат, который записал в переменную “recognized_digits”. С помощью функций библиотеки Tkinter создаем графический интерфейс для приложения (рисунок 12). В приложении будут присутствовать кнопка для загрузки

изображения, поле, отображающее выбранное изображение для распознавания и надпись с распознанным результатом.

2.5 Использование созданного приложения в образовании

Полученное приложение можно использовать в различных отраслях образования для уменьшения времени проверки работ учеников. С ее помощью можно проверять тесты, где в ответах должны стоять только цифры или последовательности цифр. Кроме этого, можно проверять бланки ответов экзаменационных листов студентов.

Например, разработанное приложение можно использовать для проверки упражнения для устного счета:

Задача 1.

Решите примеры и запишите ответы в строчку без пробелов:

12+34 =

12 - 8 =

3 + 5 =

9 - 1 =

Верным ответом будет 46488. После загрузки фотографии в приложение учитель получит результат, как на рисунке 13.

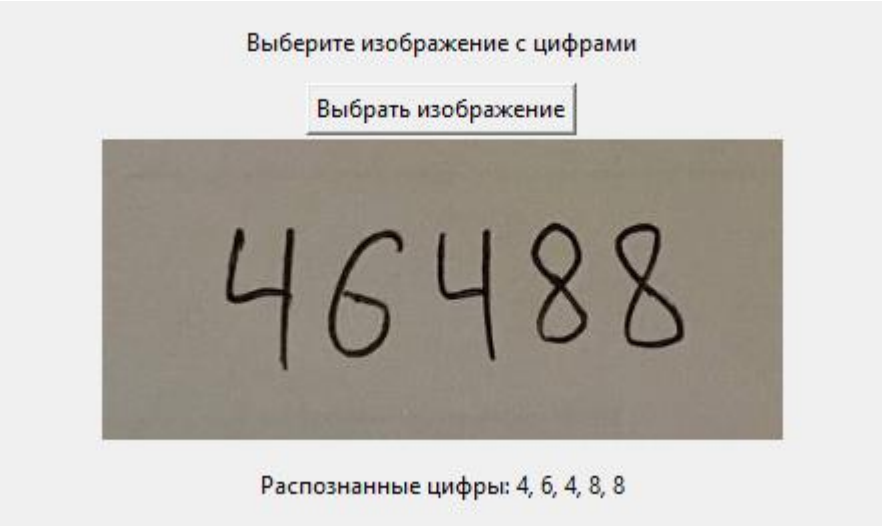

Рисунок 13 - Результат задачи 1

Задача 2.

Расставьте героев сказки “Репка” в порядке их появления в произведении:

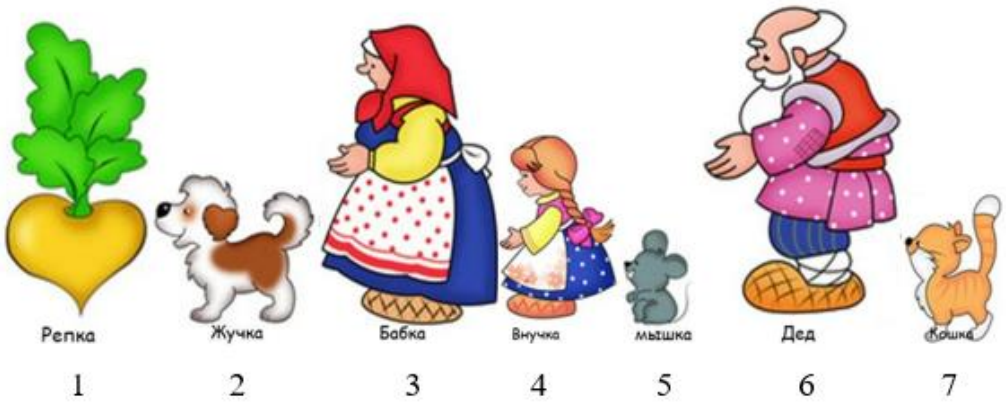

Рисунок 14 - Задача 2

Верным ответом будет 1634275. результат работы программы будет следующим:

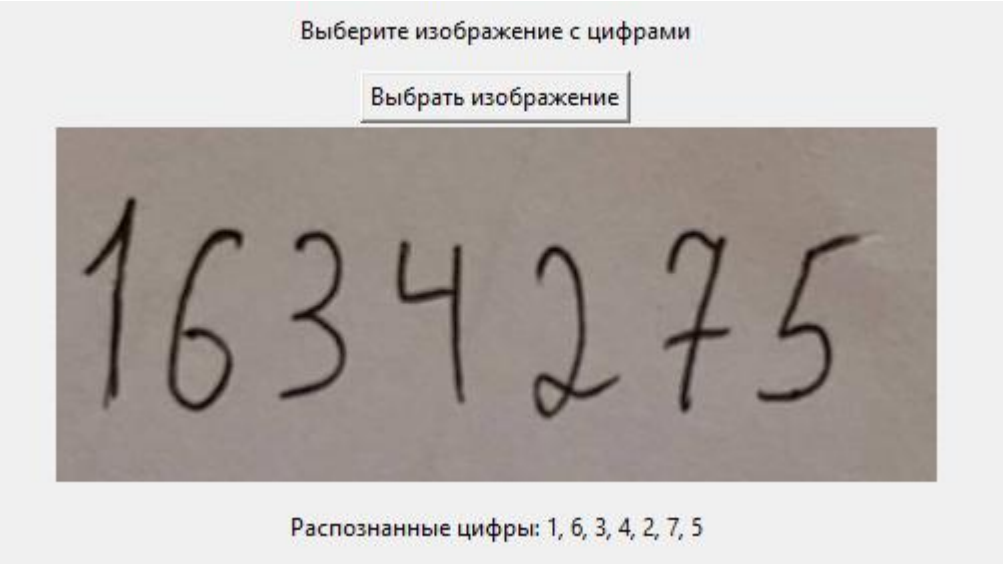

Рисунок 15 - Результат задачи 2

Разработанную НС можно использовать также для оценки правильности написания цифр учеников начальных классов. Учителя могут самостоятельно создавать тестовые наборы данных с правильно написанными цифрами и использовать нейронную сеть для сравнения написанных учениками цифр и заранее загруженными эталонными данными. В результате, преподаватель сможет оценить, насколько точно ученики сумели написать цифры и помогать не справляющимся с данной задачей ученикам.

Одним из потенциальных улучшений программы может стать функция, при которой ученики смогут сдавать свои домашние задания в приложении для автоматического распознавания ответов и отправки результатов проверки каждого ученика прямиком к учителю для оценки.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

НС - нейронная сеть;

CNN - сверточная нейронная сеть;

MNIST - Modified National Institute of Standards and Technology database;

PIL - PILLOW.

ЗАКЛЮЧЕНИЕ

В данной работе было проведено исследование возможностей применения нейронных сетей для распознавания рукописных цифр. Распознавание рукописных цифр является одной из важных задач в области компьютерного зрения, и она имеет широкий спектр применений в различных сферах, включая безопасность, медицину, автоматизацию и образование.

В работе были поставлены задачи, такие как изучение основ создания и функционирования нейронных сетей, анализ существующих методов распознавания образов, выбор структуры и инструментов для создания нейронной сети, исследование наборов данных и обучение нейронной сети для распознавания рукописных цифр.

В процессе исследования были использованы работы известных исследователей, таких как Ян Лекун, Боровикова В. П. и Джеффри Хинтон, которые представили различные методы и алгоритмы обучения нейронных сетей для распознавания образов, включая рукописные цифры. Были также проанализированы научные статьи и публикации других авторов, чтобы рассмотреть данную проблему с разных точек зрения.

В ходе работы была создана и обучена нейронная сеть с использованием базы данных MNIST, которая содержит набор изображений рукописных цифр. Нейронная сеть научилась распознавать цифры на новых изображениях путем нахождения закономерностей в изображениях и использования их для классификации. Кроме того, нейронная сеть была улучшена таким образом, позволяющим ей считывать не 1 цифру с изображения, а сразу несколько или более.

Выводы данной работы имеют как теоретическое, так и практическое значение. Теоретически, данная работа представляет обзор методов и

технологий создания и обучения нейронных сетей для распознавания образов. Практически, результаты работы могут быть использованы для создания более современных систем распознавания рукописных цифр, которые повысят эффективность и точность обработки данных в образовательных учреждениях.

В заключение, исследование нейронных сетей для распознавания рукописных цифр является актуальной и востребованной задачей в сфере компьютерного зрения. Результаты данной работы позволяют использовать нейронные сети для автоматического распознавания рукописных цифр и исследования их применимости в образовательных процессах. Дальнейшие исследования в этой области могут привести к созданию более совершенных и эффективных систем распознавания образов, что приведет к улучшению обработки данных и повышению качества образования. “Функционал современного программного обеспечения компьютерной аналитики включает тысячи тщательно тестированных аналитических процедур, постоянно пополняющихся усовершенствованными методами анализа” [4].

ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Аггарвал Ч. Нейронные сети и глубокое обучение: учебный курс.: Пер. с англ. / Ч. Аггарвал. – СПб. : Диалектика, 2020. – 752 с. – ISBN 978-5-907203-01-3

2. Андреас М. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / М. Андреас. – Москва : Альфа-книга, 2017. – 487 с.. – Текст

3. Бишоп Кристофер, М. Распознавание образов и машинное обучение / М. Бишоп Кристофер. – М. : Вильямс, 2020. – 960 с.

4. Боровиков В.П. Популярное введение в современный анализ данных и машинное обучение на Statistica / В.П. Боровиков. – Москва : Популярное введение в современный анализ данных и машинное обучение на Statistica В.П. Боровиков, 2018. – 288 с.

5. Боровикова В.П. Нейронные сети STATISTICA Neural Networks: Методология и технология современного анализа данных / В.П. Боровикова, В.П. Боровиков, Г.В. Калайдина. – Москва : Горячая линия - Телеком, 2019. – 392 с.

6. Бринк Х., Машинное обучение / Х., Бринк, Д., Ричардс, М., Феверолф. – Санкт-Петербург : Питер, 2017. – 338 с.

7. Гафаров Ф.М. Искусственные нейронные сети и их приложения: учебное пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань : Изд-во Казан. ун-та, 2018. – 121 с.

8. Емельянов, С. В. Информационные технологии и вычислительные системы. Вычислительные системы. Компьютерная графика. Распознавание образов. Математическое моделирование. Выпуск №2 / С. В. Емельянов. – Москва : Мир, 2015. – 662 с.

9. Лекун Я. Как учится машина: Революция в области нейронных сетей и глубокого обучения / Я. Лекун. – Москва : Интеллектуальная Литература, 2020. – 351 с.

10. Мюллер, А. Введение в машинное обучение с помощью Python / А. Мюллер,. – Москва : Диалектика, 2017. – 480 с.

11. Плас Д. Python для сложных задач. Наука о данных и машинное обучение. Руководство / Д. Плас. – Москва : Питер, 2018. – 759 с.

12. Потапов, А Автоматический анализ изображений и распознавание образов / А Потапов. – Москва : LAP Lambert Academic Publishing, 2017. – 292 с.

13. Рашка, С. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow / С. Рашка, В. Мирджалили. – Москва : дом Вильямс, 2019. – 770 с. – ISBN 978-1789955750.

14. Ростовцев В.С. Искусственные нейронные сети: учебник / В.С. Ростовцев. – Киров : Издательство ВятГУ, 2014. – 208 с.

15. Хеллман Д Стандартная библиотека Python 3. Справочник с примерами / Д Хеллман. – Москва; СПб : Вильямс, 2018. – 1376 с. – ISBN 978-5-6040043-8-8

16. Чару, А. Нейронные сети и глубокое обучение: учебный курс. / А. Чару,. – Санкт-Петербург : Диалектика, 2020. – 752 с.

17. Bengio, Y., Lecun, Y., & Hinton, G. Deep learning for AI Communications of the ACM, 64(7), 58-65 с.

18. Bishop, Christopher M. Bishop Pattern Recognition and Machine Learning / Christopher M. Bishop Bishop. – New York : Springer Science+Business Media, 2006. – 758 с. – ISBN 978-0387-31073-2.

19. Joseph, M. Modern Time Series Forecasting with Python - Explore industry-ready time series forecasting using modern machine learning and deep

learning / M. Joseph. – BIRMINGHAM MUMBAI : Packt Publishing, 2022. – 552 c.

20. Machine Learning / T. Michel, B. M. Бухштабер, И. С. Енюков и другие – New York : McGraw Hill, 2006. – 250 c.

21. Zaccone G. Deep Learning with TensorFlow / G. Zaccone, Md R. Karim, A Menshawy. – Birmingham : Packt Publishing, April 24, 2017. – 320 pages c. – ISBN 9781786469786