|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_« Международных образовательных программ»\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_ «Системы обработки информации и управления»\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

**«Интеллектуальная система идентификации человека по отпечаткам пальцев»\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_ИУ5, 42М\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Вей Пхьоу Ту

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_Г.И. Афанасьев

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**Г .И. Афанасьев **\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**РЕФЕРАТ**

Настоящая расчетно-пояснительная записка содержит 123 страницы, 41 рисунок, 9 таблиц, список литературы в количестве 13 наименований.

Выпускная квалификационная работа на тему "Интеллектуальная система идентификации человека по отпечаткам пальцев". Расчетно-пояснительная записка к выпускной квалификационной работе состоит из трех частей.

Предпроектное исследование содержит описание и анализ предметной области, выбор критериев для сравнения, сравнительный анализ и выбор средств реализации.

Часть конструкторская содержит обоснование выбора архитектуры системы, обоснование выбора языков программирования, обоснование выбора набора данных, обоснование выбора фреймворка, анализ и выбор моделей сверточных нейронных сетей.

В исследовательской части представлены результаты моделирования трех сверточных нейронных сетей и выбора с наилучшей точностью и с наименьшими потерями за короткое время.

В заключении содержит выводы по разработанному проекту.

Приведен список используемых источников, используемых при написании ВКРМ. Графическая содержит 22 слайдов.

**Содержание**

РЕФЕРАТ 2

CОДЕРЖАНИЕ 3

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 6

ВВЕДЕНИЕ 7

1. ПРЕДПРОЕКТНОЕ ОБСЛЕДОВАНИЕ 8

1.1. Цель и назначение разработки 8

1.2.Анализ предметной области 8

1.2.1. Распознавание лиц 10

1.2.2. Распознавание голоса 11

1.2.3. Распознавание радужной оболочки глаза 12

1.2.4. Геометрия кисти и пальцев 13

1.2.5. Проверка подписи 13

1.2.6. Отпечатки пальцев 13

1.3. Методы и подходы для идентификации человека по отпечаткам пальцев 17

1.3.1. Изображение отпечатка пальца 17

1.3.2. Получение изображений отпечатков пальцев 18

1.3.3. Обработка изображений отпечатков пальцев 18

1.3.4. Алгоритм нормализации профиля отпечатка пальца 18

1.3.5. Бинаризация отпечатков пальцев 19

1.3.6. Извлечение характеристик отпечатков пальцев 20

1.4. Нейронные сети 20

1.5. Анализ аналогов и прототипов системы 22

1.6. Постановка задачи проектирования (математическая) 23

2. КОНСТРУКТОРСКАЯ ЧАСТЬ 26

2.1. Анализ и выбор архитектуры НС 26

2.1.1. Нейронные сети прямого распространения (FFNN) 26

2.1.2. Рекуррентные нейронные сети (RNN) 30

2.1.3. Свертрчные нейронные сети (CNN) 39

2.2. Анализ и выбор прототипов НС 43

2.2.1. Базовое ядро сети CNN 44

2.3. Анализ и выбор методов обучения НС 51

2.4. Анализ и выбор методов предварительного анализа, очистки, обработки и разбиения исходных данных. 54

2.5. Анализ и выбор среды для разработки НС 59

2.5.1. PyTorch 60

2.5.2. Tensorflow 65

2.5.3. Caffe 70

2.5.4. Keras 74

2.6. Анализ и выбор функций активации 78

2.6.1. Sigmoid 78

2.6.2. Tanh 79

2.6.3. ReLU 80

2.6.4. Leaky ReLU 81

2.7. Анализ и выбор методов инициализации весов НС 81

2.7.1. Инициализация со всеми нулями 82

2.7.2. Случайная инициализация 82

2.7.3. Инициализация He 83

2.8. Анализ и выбор функции потерь НС 85

2.8.1. Регрессионные потери 85

2.8.2. Классификационные Потери 87

2.9. Анализ и выбор показателей качества (метрик) НС 88

2.10. Анализ и выбор методов оптимизации НС 89

2.11. Анализ и выбор методов регуляризации для предотвращения обучения 96

2.12. Конструирование топологии НС с применением добавления/удаления дополнительных скрытых слоев и прореживания 100

2.13. Структура и состав компонентов системы 102

2.14. Структура и состав программного обеспечения системы. 103

2.15. Интерфейс взаимодействия пользователя с системой 104

3. ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 107

3.1. Экспериментальная отработка и настройка проектных решений системы (с применением модификации числа периодов обучения, числа нейронов в скрытых слоях, размера пакета) 107

ЗАКЛЮЧЕНИЕ 117

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 118

ПРИЛОЖЕНИЕ А. Техническое задание на ВКРМ 119

ПРИЛОЖЕНИЕ Б. Графический материал 124

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Глубокое обучение: является частью более широкого семейства методов машинного обучения, основанных на искусственных нейронных сетях с обучением представлению. Обучение может проходить под наблюдением, полу-наблюдением или без присмотра.

Машинное обучение: Машинное обучение — это область исследований, посвященная пониманию и созданию методов, которые "учатся", то есть методов, которые используют данные для повышения производительности при выполнении некоторого набора задач. Это рассматривается как часть искусственного интеллекта.

ANN - Искусственная Нейронная сеть

НС - Нейронная Сеть

FFNN - Нейронная сеть прямой связи

RNN - Рекуррентная Нейронная сеть

АЕ - Автоэнкодеры

CNN - Сверточная Нейронная сеть

MSE - Среднеквадратичная ошибка

MAE - Средняя абсолютная погрешность

NIST - Национальный институт стандартов и технологий

GM - Геометрический момент

RST - Перевод шкалы вращения

FAR - Ложная Скорость Приема

FRR - Коэффициент ложного отклонения

NLP - Обработка естественного языка

# Введение

Распознавание отпечатков пальцев является хорошо известной проблемой в распознавании образов и широко используется в современных технологиях аутентификации, таких как устройства доступа в мобильных телефонах.

Предметом данной ВКРМ является исследование применимости сверточных нейронных сетей для распознавания отпечатков пальцев. Это достигается путем разработки различных сетевых архитектур для этой задачи. Отправной точкой является архитектура, известная как сверточные нейронные сети, на которой мы основываемся, включая дополнительные компоненты, а также сетевые архитектуры, основанные на архитектуре сверточные нейронные сети.

Данные для обучения и оценки сетей предоставляются в виде изображений отпечатков пальцев в оттенках серого, и мы реализуем простой алгоритм для генерации меток истинности. Чтобы оценить нашу работу, мы измеряем производительность всех реализованных моделей с помощью общих показателей для алгоритмов распознавания отпечатков пальцев. Наконец, перечислены проблемы с нашим подходом и указаны потенциальные будущие улучшения.

Понятно, что в последнее время гипотеза и использование глубокого обучения достигли невероятного прогресса. Поскольку большая часть вычислений машинного обучения нуждается в атрибутах в качестве информации, отличное изображение компонента можно рассматривать как ключ к алгоритму. Традиционное определение и определение выделения часто завершаются искусственно.

В любом случае, выбор основных моментов в руководстве регулярно утомителен, а компонент чаще всего обладает определенной субъективностью, но, кроме того, требует некоторой более ранней информации.

.

# 1. ПРЕДПРОЕКТНОЕ ОБСЛЕДОВАНИЕ

## 1.1. Цель и назначение разработки

Распознавание отпечатков пальцев является одним из наиболее известных биометрических методов, и на сегодняшний день это наиболее используемое биометрическое решение для аутентификации в компьютерных системах. Причинами такой популярности распознавания отпечатков пальцев являются простота получения, устоявшееся использование и принятие по сравнению с другими биометрическими данными, а также тот факт, что существует множество (десять) источников этой биометрии для каждого человека. Распознавание отпечатков пальцев — это процедура сравнения известных и неизвестных отпечатков пальцев, чтобы доказать, принадлежат они одному и тому же человеку или нет. Сегодня для сопоставления отпечатков пальцев и решения связанных с этим проблем используется множество подходов, методов и систем.

Одно из наиболее важных применений отпечатков пальцев - помочь следователям связать одно место преступления с другим с участием одного и того же человека. Идентификация отпечатков пальцев также помогает следователям отслеживать досье преступников, их предыдущие аресты и судимости, чтобы помочь в вынесении приговоров, условно-досрочном освобождении и принятии решений о помиловании.

Целью данной ВКРМ является синтез интеллектуальной системы идентификации человека по отпечаткам пальцев.

## 1.2. Анализ предметной области

Биометрия, которая относится к идентификации человека на основе его или ее физиологических или поведенческих характеристик. Физиологические характеристики включают изображения рук или пальцев, характеристики лица и распознавание радужной оболочки глаза. Поведенческие характеристики - это черты, которые усваиваются или приобретаются. Динамическая проверка подписи, проверка голоса и динамика нажатия клавиш.

Не существует одной “идеальной” биометрии, которая соответствовала бы всем потребностям. Все биометрические системы имеют свои преимущества и недостатки. Однако существуют некоторые общие характеристики, необходимые для использования биометрической системы.

Во-первых, биометрические данные должны основываться на различимом признаке. Существует множество научных данных, подтверждающих идею о том, что “нет двух одинаковых отпечатков пальцев”. Такие технологии, как геометрия руки, используются уже много лет, а такие технологии, как распознавание лиц или радужной оболочки глаза, получили широкое распространение. Некоторые новые биометрические методы могут быть столь же точными, но могут потребовать дополнительных исследований для установления их уникальности. Другим ключевым аспектом является то, насколько “удобна для пользователя” система. Процесс должен быть быстрым и простым, например, сделать снимок с помощью видеокамеры, поговорить в микрофон или прикоснуться к сканеру отпечатков пальцев. Низкая стоимость важна, но большинство разработчиков понимают, что речь идет не только о первоначальной стоимости датчика или соответствующего программного обеспечения. Часто затраты на поддержку жизненного цикла, связанные с предоставлением системного администрирования и оператора регистрации, могут превышать первоначальную стоимость биометрического оборудования. Преимущество биометрической аутентификации заключается в возможности требовать большего количества экземпляров аутентификации таким быстрым и простым способом, чтобы пользователи не беспокоились о дополнительных требованиях. По мере развития биометрических технологий и их широкого коммерческого использования работа с несколькими уровнями аутентификации или несколькими экземплярами аутентификации станет менее обременительной для пользователей. Биометрия широко используется в криминалистических приложениях, таких как идентификация преступников и охрана тюрем. Биометрическая технология быстро развивается и имеет очень большой потенциал для широкого применения в гражданских приложениях, таких как электронный банкинг, электронная коммерция и контроль доступа. В связи с быстрым увеличением числа и использования электронных транзакций электронное банковское дело и электронная торговля становятся одними из наиболее важных новых применений биометрии. Биометрическое распознавание может использоваться в режиме идентификации, когда биометрическая система идентифицирует человека из всего зарегистрированного населения путем поиска совпадения в базе данных исключительно на основе биометрических данных.

Различными типами методов биометрического распознавания являются:

* Распознавание лиц
* Распознавание голоса
* Распознавание радужной оболочки глаза
* Геометрия кисти и пальцев
* Проверка подписи
* Отпечатки пальцев

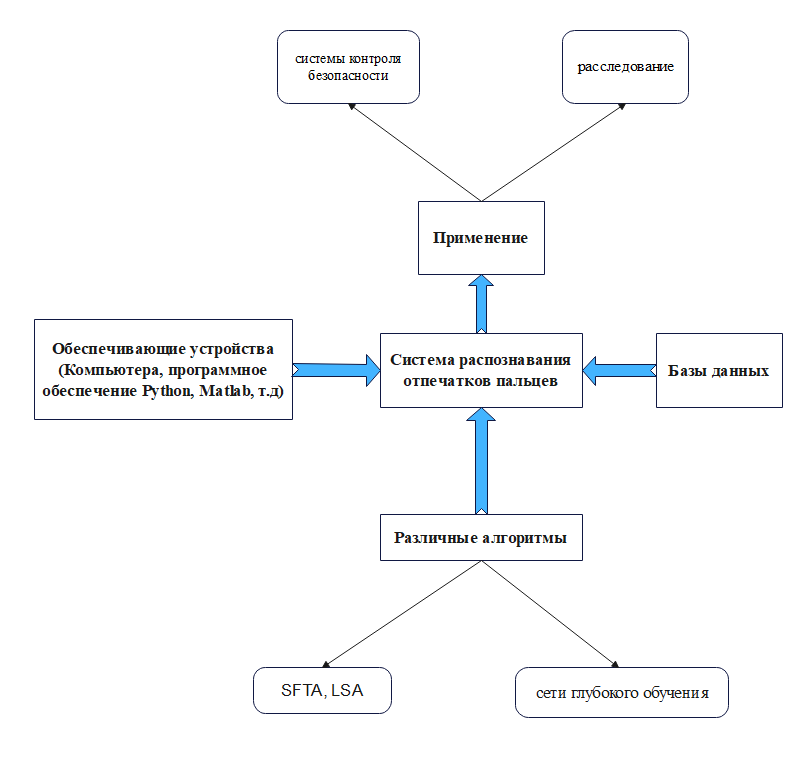


Рисунок 1.1. Описание предметной области

### 1.2.1. Распознавание лиц

Идентификация человека по изображению его лица может быть выполнена несколькими различными способами, например, путем захвата изображения лица в видимом спектре с помощью недорогой камеры или с помощью инфракрасных моделей теплового излучения лица. Распознавание лиц в видимом свете обычно моделирует ключевые особенности из центральной части изображения лица. Используя широкий ассортимент камер, системы видимого света извлекают из захваченного изображения (изображений) черты, которые не меняются с течением времени, избегая при этом поверхностных черт, таких как выражение лица или волосы. Несколько подходов к моделированию изображений лиц в видимом спектре включают анализ основных компонентов, анализ локальных признаков, нейронные сети, теорию упругих графов и анализ с несколькими разрешениями.

Некоторые из проблем, связанных с распознаванием лиц в визуальном спектре, включают уменьшение воздействия переменного освещения и обнаружение маски или фотографии. Некоторым системам распознавания лиц может потребоваться неподвижный или позирующий пользователь для захвата изображения, хотя многие системы используют процесс в реальном времени для обнаружения головы человека и автоматического определения местоположения лица. Основные преимущества распознавания лиц заключаются в том, что оно является ненавязчивым, громкой связью, непрерывным и принимается большинством пользователей [3].

### 1.2.2. Распознавание голоса

История распознавания голоса насчитывает около четырех десятилетий, когда выходные данные нескольких аналоговых фильтров усреднялись по времени для сопоставления. Распознавание голоса использует акустические особенности речи, которые, как было обнаружено, различаются у разных людей. Эти акустические паттерны отражают как анатомию (например, размер и форму горла и рта), так и усвоенные поведенческие паттерны (например, высоту голоса, стиль речи). Это включение выученных шаблонов в голосовые шаблоны (последние называются "голосовые отпечатки") привело к тому, что система распознавания речи была классифицирована как "поведенческая биометрическая". Системы распознавания голоса используют три стиля устного ввода: зависящий от текста, текстовый запрос и независимый от текста. Большинство приложений для голосовой проверки используют текстовый ввод, который включает в себя выбор и ввод одного или нескольких голосовых паролей. Ввод с текстовым запросом используется всякий раз, когда есть опасения по поводу самозванцев. Различные технологии, используемые для обработки и хранения голосовых отпечатков, включают скрытые марковские модели, алгоритмы сопоставления с образцом, нейронные сети, матричное представление и деревья решений. Снижение производительности может быть вызвано изменениями поведенческих характеристик голоса, а также регистрацией с использованием одного телефона и проверкой по другому телефону. Системы распознавания также должны учитывать изменения голоса, вызванные старением. Многие компании продают системы распознавания голоса, часто как часть крупных систем обработки голоса, управления и коммутации. Захват биометрических данных рассматривается как неинвазивный. Технология не требует дополнительного оборудования за счет использования существующих микрофонов и технологии передачи голоса, позволяющей распознавать на больших расстояниях с помощью обычных телефонов (проводных или беспроводных) [5].

### 1.2.3. Распознавание радужной оболочки глаза

Этот метод распознавания использует радужную оболочку глаза, которая представляет собой цветную область, окружающую зрачок. Узоры радужной оболочки считаются уникальными. Узоры радужной оболочки получаются с помощью системы получения изображений на основе видео. Устройства для сканирования радужной оболочки глаза уже несколько лет используются в приложениях для аутентификации личности. Системы, основанные на распознавании радужной оболочки глаза, существенно подешевели, и ожидается, что эта тенденция сохранится. Технология хорошо работает как в режиме проверки, так и в режиме идентификации (в системах, выполняющих поиск по принципу "один ко многим" в базе данных). Современные системы можно использовать даже при наличии очков и контактных линз. Технология не является навязчивой. Это не требует физического контакта со сканером. Было продемонстрировано, что распознавание радужной оболочки глаза работает с людьми из разных этнических групп и национальностей [6].

### 1.2.4. Геометрия кисти и пальцев

Эти методы личной аутентификации хорошо зарекомендовали себя. Распознавание рук доступно уже более двадцати лет. Для достижения личной аутентификации система может измерять либо физические характеристики пальцев, либо рук. К ним относятся длина, ширина, толщина и площадь поверхности кисти. Одной интересной особенностью является то, что некоторые системы требуют небольшой биометрической выборки (несколько байт). Геометрия руки получила признание в самых разных областях применения. Его часто можно найти в физическом контроле доступа в коммерческих и жилых приложениях, в системах учета рабочего времени и посещаемости, а также в обычных приложениях для аутентификации пользователей [7].

### 1.2.5. Проверка подписи

Эта технология использует динамический анализ подписи для аутентификации человека. Технология основана на измерении скорости, давления и угла, используемых человеком при создании подписи. Одним из направлений применения этой технологии были приложения для электронного бизнеса и другие приложения, где подпись является общепринятым методом личной аутентификации [4].

### 1.2.6. Отпечатки пальцев

Узоры фрикционных выступов и впадин на кончиках пальцев человека уникальны для этого человека. На протяжении десятилетий правоохранительные органы классифицировали и определяли личность, сопоставляя ключевые точки окончаний хребтов и бифуркаций. Отпечатки пальцев уникальны для каждого пальца человека, включая идентичных близнецов. Одна из наиболее коммерчески доступных биометрических технологий, устройства распознавания отпечатков пальцев для доступа к настольным компьютерам и ноутбукам теперь широко доступны, пользователям больше не нужно вводить пароли – вместо этого мгновенный доступ обеспечивает только касание. Системы отпечатков пальцев также могут использоваться в режиме идентификации. Несколько штатов проверяют отпечатки пальцев у новых претендентов на получение пособий по социальному обеспечению, чтобы убедиться, что получатели не получают пособия обманным путем под вымышленными именами [9].

Отпечатки пальцев представляют собой Рисунки гребней и борозд на кончике пальца и широко используются для идентификации личности людей. Биологические свойства формирования отпечатков пальцев хорошо изучены, и отпечатки пальцев веками использовались для целей идентификации. С начала 20-го века отпечатки пальцев широко использовались для идентификации преступников различными судебно-медицинскими отделами по всему миру. Из-за его криминального подтекста некоторые люди чувствуют себя некомфортно, предоставляя свои отпечатки пальцев для идентификации в гражданских приложениях. Однако, поскольку биометрические системы на основе отпечатков пальцев обеспечивают положительную идентификацию с очень высокой степенью достоверности, а компактные твердотельные датчики отпечатков пальцев могут быть встроены в различные системы (например, сотовые телефоны), аутентификация на основе отпечатков пальцев становится все более популярной в ряде гражданских и коммерческих приложений, таких как, социальное обеспечение выплата, доступ к сотовому телефону и вход в систему с портативного компьютера. Доступность дешевых и компактных твердотельных сканеров, а также надежных устройств сопоставления отпечатков пальцев являются двумя важными факторами популярности систем идентификации на основе отпечатков пальцев. Отпечатки пальцев также имеют ряд недостатков по сравнению с другими биометрическими данными [2].

Таблица 1. Преимущества и недостатки биометрических методов

|  |  |  |
| --- | --- | --- |
| **Метод** | **Преимущества** | **Недостатки** |
| Проверка отпечатков пальцев | * Высокая надежность * Надежные * Очень отличительный * Проверенная точность * Передовые технологии * Удобство для пользователя * Уникальность * Стабильная с течением времени | * Травма может повлиять * Сухая кожа может вызвать трудности |
| Геометрия руки | * Небольшой Шаблон * Не зависит от состояния кожи | * Размер сканера * Травма может повлиять * Низкая различимость |
| Распознавание лиц | * Эффективный Процесс * Высокое признание | * Изменение лица с течением времени * Можно манипулировать хирургическим путем * Невозможно отличить близнецов * Религиозные или культурные запреты * Плохая экология |
| Сканирование радужной оболочки глаза | * Уникальность * Надежные * Очень отличительный | * Сложный процессор * Высокая Стоимость * Плохая экология * Относительно новая технология * Страдающий диабетом |
| Распознавание голоса | * Высокий уровень принятия пользователями * Высокое признание * Низкие требования к обучению | * Голос и язык меняются со временем * Легко манипулировать * Низкая Точность * Плохая экология * Грипп или инфекция горла |
| Распознавание подписи | * Высокое признание пользователей * Низкие требования к обучению | * Нестабильный с течением времени * Изменения с течением времени * Низкая различимость |

## 1.3. Методы и подходы для идентификации человека по отпечаткам пальцев

Теперь я расскажу о методах и подходах к идентификации отпечатков пальцев. Использование отпечатков пальцев в биометрической идентификации было наиболее широко используемой системой аутентификации. Уникальность отпечатка пальца для каждого человека дает нам все необходимое для безупречной идентификации. Однако в процессе сканирования отпечатков пальцев изображение, генерируемое сканером, может немного отличаться во время каждого сканирования. В этой статье рассматривается реализация искусственных нейронных сетей для обеспечения эффективного алгоритма сопоставления для аутентификации по отпечаткам пальцев.

### 1.3.1. Изображение отпечатка пальца

Отпечаток пальца — это характерный Рисунок одного пальца. У каждого человека есть свои собственные отпечатки пальцев с постоянной уникальностью [7]. Благодаря уникальности и постоянству отпечатков пальцев, они являются одними из самых надежных характе Ристик человека, которые могут быть использованы для идентификации личности. Отпечаток пальца — это графический рисунок из гребней и впадин на поверхности человеческого пальца. Благодаря уникальности и постоянству отпечатков пальцев, они являются одними из самых надежных характеристик человека, которые могут быть использованы для идентификации личности. Производительность автоматической системы идентификации отпечатков пальцев в значительной степени зависит от качества изображения отпечатков пальцев, на которое может влиять несколько факторов, таких как наличие шрамов, колебания давления между пальцем и датчиком сбора, изношенные артефакты и условия окружающей среды в процессе сбора.

### 1.3.2. Получение изображений отпечатков пальцев

Для получения отпечатков пальцев используется целый ряд методов. Среди них есть метод оттиска чернил, который остается самым популярным. Также пРисутствуют имеются сканеры отпечатков пальцев без чернил, устраняющие промежуточный процесс оцифровки. Качество отпечатков пальцев очень важно, поскольку оно напрямую влияет на алгоритм извлечения деталей. Размер отсканированных отпечатков пальцев, которые используются в этом исследовании, составляет 188 x 240 пикселей. Изображения делаются в таком размере, чтобы облегчить вычислительную нагрузку.

### 1.3.3. Обработка изображений отпечатков пальцев

Обработка изображения отпечатка пальца необходима для: улучшения четкости структур гребней изображений отпечатков пальцев; сохранения их целостности; предотвращения введения ложных структур или артефактов; сохранения связности гребней при сохранении разделения между гребнями. Операции обработки изображений отпечатков пальцев – это операции по улучшению изображений, нормализация изображений и бинаризация изображений [8].

### 1.3.4. Алгоритм нормализации профиля отпечатка пальца

Идеально воспринимаемое или отсканированное изображение отпечатка пальца имеет четкие и отчетливые выступы и впадины. Например, идеальным изображением отпечатка пальца может быть свернутый оттиск чернил на карточке отпечатка пальца. Профиль кожи пальца, выполненный из гребней, равномерно нанесен на плоскую бумажную карточку, которая оставляет чернильный отпечаток гребней в виде непрерывного потока проходов переднего плана, а между гребнями находятся долины в качестве фона белого цвета. Но на самом деле устройства для сканирования отпечатков пальцев далеки от этой идеальной настройки. Даже изображения отпечатков пальцев Национального института стандартов и технологий (NIST), отсканированные с подписанных карточек отпечатков пальцев, не идеальны. Хотя шум, вносимый в изображения отпечатков пальцев во время сканирования, является проблемой при извлечении деталей отпечатков пальцев, неравномерное распределение уровней пикселей на изображениях отпечатков пальцев также является еще одной распространенной проблемой для определения ориентации гребня. На Рисунке 1. показан пример изображения неровного профиля гребня.



Рисунок 1.2. Пример изображения отпечатка пальца с неровными профилями гребней

Существующие алгоритмы для нормализации изображений позволяют уменьшить динамический диапазон шкалы уровня серого цвета между гребнями и впадинами изображения отпечатка пальца.

### 1.3.5. Бинаризация отпечатков пальцев

Бинаризация — это процесс преобразования изображения в масштабе оттенков серого цвета в двоичное изображение, который называется бинаризацией. Нули и единицы образуют двоичное изображение. Глобальный пороговый алгоритм используется для выполнения процесса бинаризации. Просматривая каждый пиксель на изображении отпечатка пальца и решая, следует ли его преобразовать в черный (0) или белый (255), т.е. в 0 или 1 от уровня серого до черно-белого изображения, каждое значение пикселя сравнивается с пороговым уровнем, затем значение пикселя устанавливается равным нулю; в противном случае оно устанавливается равным 255 [10]. На рисунке 2 показано двоичное изображение из алгоритма бинаризации и выбранные пикселы гребня, а остальные пиксели заполнены белым цветом.



Рисунок 1.3. Изображение справа исходное изображение отпечатка пальца, слева тоже самое изображение после бинаризации

### 1.3.6. Извлечение характеристик отпечатков пальцев

Инварианты моментов являются важными дескрипторами формы в компьютерном зрении. Существует два типа дескрипторов формы: дескрипторы формы на основе контуров и дескрипторы формы на основе областей. Регулярные инварианты моментов являются одним из самых популярных и широко используемых дескрипторов формы на основе контуров [7,8]. Система компьютерного зрения, распознающая объекты на захваченных изображениях, использует Geometric Moment (GM). GM был выведен из теории алгебраических инвариантов. Метод GM был выбран для извлечения объектов изображения, поскольку генерируемые объекты являются инвариантными к преобразованию масштаба вращения (RST). GM был успешно применен так же для идентификации воздушных судов, классификации текстур и сопоставления радиолокационных изображений с оптическими изображениями.

## 1.4. Нейронные сети

Нейронная сеть — это вычислительная структура, вдохновленная изучением биологической нейронной обработки. Существует множество различных типов нейронных сетей, от относительно простых до очень сложных, так же как существует множество теорий о том, как работает биологическая нейронная обработка. Многоуровневая нейронная сеть с прямой связью имеет слои или подгруппы обрабатывающих элементов. Слой элементов обработки выполняет независимые вычисления на основе данных, которые он получает, и передает результаты на другой уровень. Следующий слой, в свою очередь, может выполнять свои независимые вычисления и передавать результаты еще одному слою. Наконец, подгруппа из одного или нескольких обрабатывающих элементов определяет выходные данные из сети. Каждый элемент обработки производит свои вычисления на основе взвешенной суммы своих входных данных. Первый слой — это входной слой, а последний - выходной слой. Слои, расположенные между первым и последним слоями, являются скрытыми слоями. Обрабатывающие элементы рассматриваются как единицы, похожие на нейроны в человеческом мозге, и, следовательно, их называют клетками, нейромимами или искусственными нейронами. Пороговая функция иногда используется для определения выходного сигнала нейрона в выходном слое. Синапсы между нейронами называются соединениями, которые представлены ребрами ориентированного графа, в котором узлами являются искусственные нейроны. Сети состоят из небольших блоков, называемых ячейками, и они соединены друг с другом таким образом, что могут передавать сигналы друг другу.

Веса, используемые для связей между различными слоями, имеют большое значение для работы нейронной сети и характеристики сети. В нейронной сети возможны следующие действия:

* Начните с одного набора весов и запустите сеть. (Без обучения)
* Начните с одного набора весов, запустите сеть, измените некоторые или все веса и снова запустите сеть с новым набором весов. Повторяйте этот процесс до тех пор, пока не будет достигнута какая-то заранее определенная цель. (Обучение)

Соединения имеют определенную прочность или вес. Сеть начинается с этих сильных сторон соединения, установленных случайным образом. Сеть подвергается воздействию различных входных данных, и сильные стороны корректируют их в соответствии с некоторым математическим планом. Это то, что мы называем обучением, и после этого сеть может распознавать шаблоны ввода или, по крайней мере, делать что-то разумное - все, чему ее обучили. Таким образом, информация хранится в сильных сторонах связей, точно так же, как и в человеческом мозге.

## 1.5. Анализ аналогов и прототипов системы

Дактилоскопия (распознавание отпечатков пальцев) является наиболее развитым биометрическим методом идентификации личности на сегодняшний день. Катализатором развития этого метода стало его широкое применение в судебной медицине в 20 веке.

У каждого человека есть уникальный папиллярный Рисунок отпечатков пальцев, который делает возможной идентификацию. Как правило, алгоритмы используют характерные точки на отпечатках пальцев: конец линии Рисунка, разветвление линии, отдельные точки. Дополнительно задействована информация о морфологической структуре отпечатка пальца: взаимное расположение замкнутых линий папиллярного Рисунка, "дугообразных" и спиральных линий. Особенности папиллярного Рисунка преобразуются в уникальный код, который сохраняет информационное содержание печатного изображения. И именно "коды отпечатков пальцев" хранятся в базе данных, используемой для поиска и сравнения. Время преобразования изображения отпечатка пальца в код и его идентификации обычно не превышает 1 с, в зависимости от размера базы. Время, затраченное на поднятие руки, не учитывается.

В качестве источника данных для FAR и FRR используются статистические данные VeriFinger SDK, полученные с использованием U.S.Использовался сканер отпечатков пальцев U DP. За последние 5-10 лет характеристики распознавания по пальцу не сильно продвинулись вперед, поэтому приведенные цифры показывают хорошее среднее значение современных алгоритмов. Сам алгоритм VeriFinger уже несколько лет побеждает в Международном конкурсе по проверке отпечатков пальцев, где соревнуются алгоритмы распознавания отпечатков пальцев.

Таблица 2. процентная доля FAR и FRR на 300 человек

|  |  |
| --- | --- |
| FAR | FRR |
| 0,10% | 0,30% |
| 0,01% | 0,40% |
| 0,00% | 0,60% |
| 0,00% | 0,90% |

Типичное значение FAR для метода распознавания отпечатков пальцев составляет 0,001%. Из таблицы мы получаем, что стабильная работа системы идентификации при FAR = 0,001% возможна при количестве 300 человек.

**Преимущества**

Высокая надежность - статистические показатели метода лучше, чем у методов идентификации по лицу, голосу, окраске. Недорогие устройства, сканирующие изображение отпечатка пальца. Довольно простая процедура сканирования отпечатка пальца.

**Недостатки**

Папиллярный Рисунок отпечатка пальца очень легко повреждается мелкими царапинами, порезами. Многие сканеры неадекватно обрабатывают сухую кожу и не пропускают пожилых людей.

## 1.6. Постановка задачи проектирования (математическая)

Пусть *I* - набор изображений журнала, *F* - набор уникальных идентификаторов изображений, называемых отпечатками пальцев, а fs и fs0 - набор отпечатков пальцев, сгенерированных и зарегистрированных в базе данных. Специальный нулевой идентификатор *f0 F* идентифицирует неизвестное изображение. Набор степеней обозначается *P (F).* Функциональные требования задачи могут быть формализованы с помощью трех функций:

(1.1) (1.2) (1.3)

Где,

(1.4) (1.5)

Есть два географически разделенных участка, на которых делаются снимки журнала *I* участок сбора и обработки. На месте сбора урожая *generate\_fingerprint* генерирует уникальный отпечаток изображения журнала, вносящий вклад в набор сгенерированных отпечатков пальцев *fs*, подмножество всех возможных таких отпечатков пальцев *F*. На производственной площадке функция *identify\_fingerprint* идентифицирует журнал на основе нового изображения того же журнала или возвращает значение *f0*, если соответствующий отпечаток пальца ранее не был создан и добавлен в *fs*. Из-за разных изображений, сделанных с одного и того же бревна на двух участках, а также из-за старения и возможных повреждений бревен при транспортировке функции идентификации и фильтрации (1.2) и (1.3) являются вероятностными, т.е. (1.4) и (1.5) выполняются только с определенными вероятностями <1.

Можно вложить несколько функций фильтра *filter\_fingerprint* (фактически их реализации). Каждая функция *filter\_fingerprint* принимает набор отпечатков пальцев и возвращает уменьшенный набор отпечатков пальцев с требованием, чтобы использование уменьшенного набора в качестве входных данных для функции *identify\_fingerprint* давало тот же результат, что и предоставление начального набора отпечатков пальцев. Таким образом, каждый из них уменьшает размер проблемы для *identify\_fingerprint*. Пусть [filter1, …, filtern] будет последовательностью или каскадом реализаций *filter\_fingerprint*. Соответствующий процесс идентификации на месте обработки определяется составной функцией:

(1.5) (1.6)

Обратите внимание, что identification\_process имеет ту же подпись, что и *identify\_fingerprint* (1.2), и соответствует его требованиям (1.4).

Для достижения цели оцениваются следующие подходы:

* Разделение набора сгенерированных отпечатков пальцев на части журналов
* Фильтрация набора отпечатков пальцев журнала до фактической идентификации с использованием детерминированных и основанных на нейронных сетях реализаций функции *filter\_fingerprint* (каскад фильтров [filter1, ..., filtern])
* Комбинация секционирования и фильтрации.

# 2. КОНСТРУКТОРСКАЯ ЧАСТЬ

## 2.1. Анализ и выбор архитектуры НС

Искусственная нейронная сеть (ANN) — это парадигма обработки информации, вдохновленная мозгом, как и люди, учатся на примерах. ANN настраивается для конкретного приложения, такого как распознавание образов или классификация данных, в процессе обучения. Обучение в значительной степени включает в себя корректировку синаптических связей, существующих между нейронами.

### 2.1.1. Нейронные сети прямого распространения (FFNN)

Нейронная сеть с прямой связью — это искусственная нейронная сеть, в которой соединения между узлами не образуют цикла. Во время этой сети информация перемещается исключительно в одном направлении и проходит через совершенно разные уровни, чтобы североамериканские страны могли использовать выходной уровень. Он проходит через входной слой, за которым следует скрытый слой, и так далее к выходному слою, где бы мы ни стремились получить желаемый результат. Эта область в основном используется для контролируемого обучения везде, где у нас есть тенденция уже воспринимать требуемую работу.

**Приложения нейронной сети с прямой связью**

Эти нейронные сети используются для многих приложений. Некоторые из них упоминаются следующим образом.

* Физиологическая система прямой связи: во время этого управление прямой связью воплощается в обычной превентивной регуляции сердцебиения перед тренировкой с помощью центрального непроизвольного
* Регуляция генов и прямая связь: во время этого мотив, по-видимому, преобладает в прославленных сетях, и было показано, что этот мотив является системой прямой связи для обнаружения временной модификации атмосферы.
* Автоматизация и управление машинами: прямое управление может быть дисциплиной среди средств автоматизации, используемых в
* Параллельная компенсация прямой связи с производной: Это довольно новый метод, который изменяет часть системы передачи с разомкнутым контуром системы с минимальными деталями на минимальную часть.

Основная причина для прямой сети заключается в приближенной работе. Если мы склонны добавлять обратную связь от последнего скрытого слоя к основному скрытому слою, это будет представлять собой повторяющуюся нейронную сеть.

Нейронная сеть с прямой связью дополнительно называется многослойным персептроном. Это сеть, в которой ориентированный граф, устанавливающий взаимосвязи, не имеет замкнутых путей или петель. Эти сети обладают жизненно важными технологическими возможностями, однако не имеют внутренней динамики.

Для создания нейронной сети с прямой связью нам нужны некоторые части, которые area unit использовала для разработки алгоритмов.

* Оптимизатор - для ослабления действия значения используется оптимизатор; он обновляет значения весов и смещений один раз в каждом цикле обучения, пока значение не достигнет всего мира.
* Стохастический градиентный спуск: это неизменная методология оптимизации целевой операции с соответствующими свойствами плавности.
* Adagrad
* Адам
* Среднеквадратичная опора

Это оптимизационное алгоритмическое правило имеет 2 формы алгоритмов;

* Алгоритм оптимизации первого порядка - Эта производная от первой производной сообщает стране Северной Америки, уменьшается или увеличивается функция для выбранной цели. Он обеспечивает дорогу, которая проходит по касательной к поверхности.
* Алгоритм оптимизации второго порядка - Этот побочный продукт второго порядка обеспечивает североамериканскую страну квадратичной поверхностью, которая касается кривизны поверхности ошибки.

**Функция затрат:**

Анализ затрат может быть трудно визуализировать; каким бы умным ни была нейронная сеть в отношении ее обучения, а также ожидаемого результата. Это даже будет зависеть от весов, а также от предубеждений.

Некоторые функции с двойным значением являются:

* Квадратичное значение
* Значение кросс-энтропии
* Экспоненциальное значение
* Расстояние Хеллингера

Он должен удовлетворять 2 свойствам для работы со значением. Они являются:

* Значение operate должно быть записано в виде медианы.
* Значение operate не должно вызывать энтузиазма по поводу какой-либо стоимости активации сети рядом с выходным уровнем.

**Архитектура нейронной сети прямого распространения**

Архитектура нейронной сети с прямой связью показана ниже:

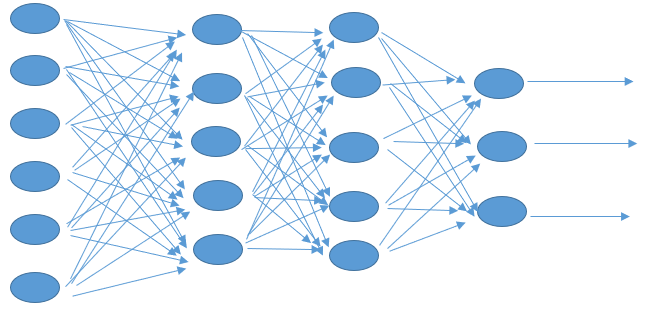


Рисунок 2.1. Архитектура нейронной сети прямого распространения

Верхняя часть Рисунка представляет собой конструкцию многослойной нейронной сети с прямой связью. Он представляет скрытые слои, а также скрытую единицу каждого слоя от входного слоя до выходного слоя.

Работа скрытых нейронов заключается в том, чтобы вмешиваться между входной и выходной сетью. Область статистики верхнего порядка извлекается путем добавления большого количества скрытых слоев в сеть.

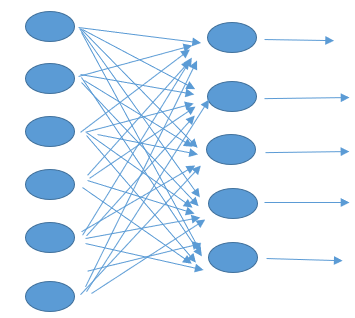


Рисунок 2.2. Однослойная (простая) нейронная сеть с прямым подключением

На верхнем рисунке представлена однослойная нейронная спецификация с прямой связью. Во время этого входные данные передаются на выходной уровень через веса и нейроны в выходном слое для вычисления выходных сигналов.

Каждый нейрон сети полностью подключен ко всем нейронам слудующего уровня. Необходимой особенностью нейронной сети является то, что она отличает ее от традиционного ПК, — это ее способность к обучению.

Выводы.

На основании вышеизложенно можно сделать вывод сети FFNN не позволяют учесть пространственные характеристики изображений отпечатков пальцев, поэтому они не подходят для цели проектирования

### 2.1.2. Рекуррентные нейронные сети (RNN)

Рекуррентная нейронная сеть является одним из типов искусственных нейронных сетей (ANN) и используется в прикладных областях обработки естественного языка (NLP) и распознавания речи. Модель RNN предназначена для распознавания последовательных характеристик данных и последующего использования шаблонов для прогнозирования предстоящего сценария.

**Работа рекуррентных нейронных сетей**

Когда мы говорим о традиционных нейронных сетях, все выходы и входы независимы друг от друга, как показано на приведенной ниже диаграмме:

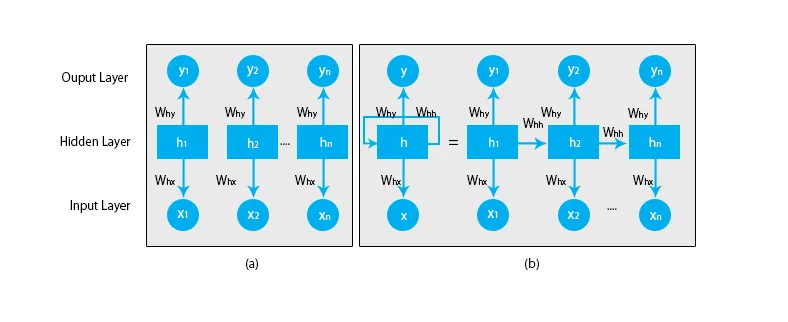


Рисунок 2.3. (a) FFNN, (b) -RNN

Но в случае рекуррентных нейронных сетей выходные данные предыдущих шагов подаются на вход текущего состояния. Например, чтобы предсказать следующую букву любого слова или предсказать следующее слово предложения, необходимо запомнить предыдущие буквы или слова и сохранить их в той или иной форме памяти.

Скрытый слой — это тот, который запоминает некоторую информацию о последовательности. Простой пример из реальной жизни, с которым мы можем связать RNN, - это когда мы смотрим фильм, и во многих случаях мы в состоянии предсказать, что произойдет дальше, но что, если кто-то только что пРисоединился к фильму, и его просят предсказать, что произойдет дальше? Каков будет его ответ? Он или она не будут иметь ни малейшего представления, потому что они не знают о предыдущих событиях фильма, и у них нет никаких воспоминаний об этом.

Ниже приведена иллюстрация типичной модели RNN:

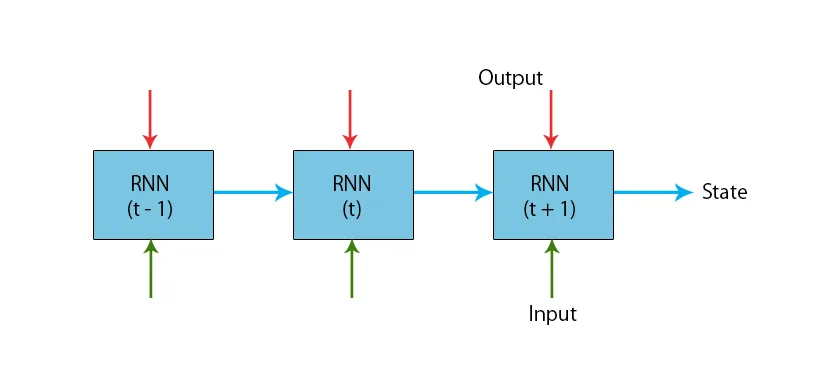


Рисунок 2.4. Типичная модель RNN

Модели RNN имеют память, которая всегда запоминает, что было сделано на предыдущих этапах и что было рассчитано. Одна и та же задача выполняется на всех входах, и RNN использует один и тот же параметр для каждого из входов. Поскольку традиционная нейронная сеть имеет независимые наборы входных и выходных данных, они более сложны, чем RNN.

У нас есть нейронная сеть с 1 входным слоем, 3 скрытыми слоями и 1 выходным слоем.

Когда мы говорим о других или традиционных нейронных сетях, они будут иметь свои собственные наборы смещений и весов в своих скрытых слоях, таких как (w1, b1) для скрытого слоя 1, (w2, b2) для скрытого слоя 2 и (w3, b3) для третьего скрытого слоя, где:w1, w2 и w3 - это веса, а b1, b2 и b3 - смещения.

Учитывая это, мы можем сказать, что каждый слой не зависит ни от какого другого и что они ничего не могут вспомнить о предыдущем вводе:

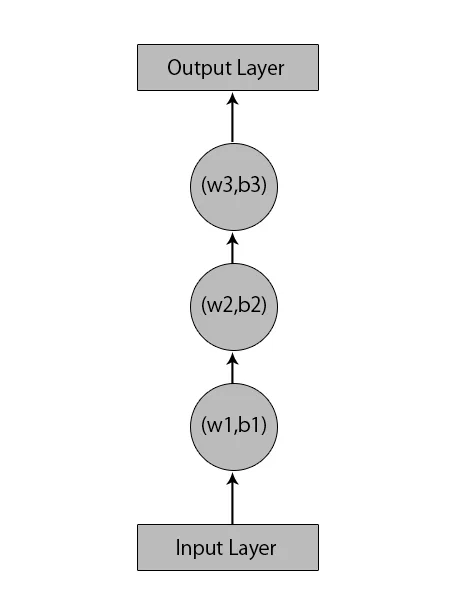


Рисунок 2.5. FFNN

Теперь, что будет делать RNN, так это следующее:

* Независимые слои будут преобразованы в зависимый слой. Это делается путем предоставления одинаковых смещений и весов для всех слоев. Это также уменьшает количество параметров и слоев в рекуррентной нейронной сети, и это помогает RNN запоминать предыдущие выходные данные, выводя предыдущие выходные данные в качестве входных данных для предстоящего скрытого слоя.
* Подводя итог, все скрытые слои могут быть объединены в один повторяющийся слой таким образом, чтобы веса и смещение были одинаковыми для всех скрытых слоев.

Таким образом, рекуррентная нейронная сеть будет выглядеть примерно так, как показано ниже:

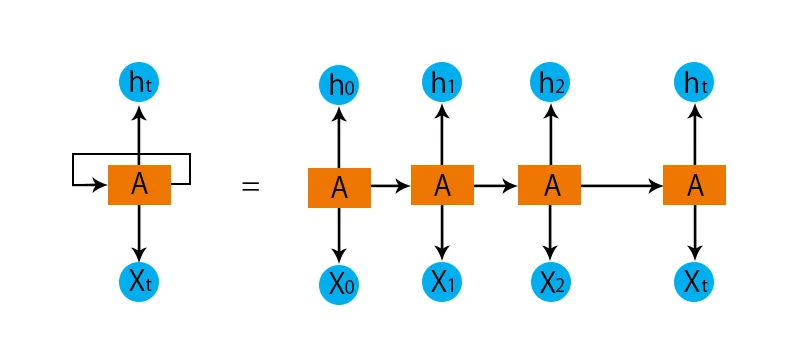


Рисунок 2.5.

**Шаги для обучения рекуррентной нейронной сети**

Ниже приведены несколько шагов для обучения рекуррентной нейронной сети.

* Во входных слоях исходные входные данные отправляются со всеми, имеющими одинаковый вес и функцию активации.
* Используя текущий ввод и вывод предыдущего состояния, вычисляется текущее состояние.
* Теперь текущее состояние ht станет ht-1 для второго временного шага.
* Это продолжает повторять все шаги, и для решения какой-либо конкретной проблемы это может продолжаться столько раз, чтобы объединить информацию со всех предыдущих шагов.
* Затем последний шаг вычисляется по текущему состоянию конечного состояния и всем другим предыдущим шагам.
* Теперь ошибка генерируется при вычислении разницы между фактическим результатом и результатом, сгенерированным нашей моделью RNN.
* Заключительный шаг — это когда происходит процесс обратного распространения, в котором ошибка передается в обратном направлении для обновления весов.

**Преимущества**

* RNN может обрабатывать входные данные любой длины.
* Модель RNN моделируется для запоминания каждой информации в течение всего времени, что очень полезно в любом предсказателе временных рядов.
* Даже если размер входных данных больше, размер модели не увеличивается.
* Веса могут быть разделены между временными шагами.
* RNN могут использовать свою внутреннюю память для обработки произвольной серии входных данных, чего нельзя сказать о нейронных сетях с прямой связью.

**Недостатки**

* Из-за его повторяющегося характера вычисления выполняются медленно.
* Обучение моделей RNN может быть затруднено.
* Если мы используем relu или tanh в качестве функций активации, становится очень сложно обрабатывать очень длинные последовательности.
* Подвержен таким проблемам, как взрыв и исчезновение градиента.

**Вывод**

На основании вышеизложенно можно сделать вывод сети RNN не позволяют учесть пространственные характеристики изображений отпечатков пальцев, поэтому они не подходят для цели проектирования.

### 2.1.3. Свертрчные нейронные сети (CNN)

### 

Сверточные нейронные сети используются для образов изображений, в то время как они принимают входные данные и отличают выходные данные от противоположных. Это используется в таких приложениях, как классификация изображений и анализ медицинских изображений. Это упорядоченная версия многослойного персептрона, который представляет собой один слой вегетативной клетки, соединенный со следующим слоем.

Сверточная нейронная сеть состоит из входного слоя ассоциированной степени, ассоциированного с выходным слоем, дополнительно в виде нескольких скрытых слоев. Скрытые слои CNN обычно содержат ряд сверточных слоев, которые скручиваются при умножении или альтернативном действительном числе.

Сверточный слой внутри нейронной сети должен иметь следующие атрибуты:

* Сверточные ядра, очерченные размером
* Разнообразие входных и выходных каналов
* Глубина сверточного фильтра должна соответствовать количеству каналов входной карты объектов.

**Особенности сверточной нейронной сети**

Сверточные нейронные сети обладают следующими характерными особенностями:

* Слои сверточной нейронной сети состоят из нейронов, организованных по трем категориям: вес, высота и глубина.
* Локальное свойство: Они используют пространственный раздел, реализуя шаблон связи по соседству между нейронами соседних слоев.
* Общие веса: каждый фильтр реплицируется по всему визуальному представлению.
* Объединение: во время этих слоев объединения карты объектов делятся на прямоугольные подобласти, и поэтому объект в каждом параллелограмме по отдельности отбирается с понижением до одной цены путем взятия их средней цены.

**Дополнительные Гиперпараметры**

Сверточные нейронные сети используют дополнительные гиперпараметры, чем обычный многослойный персептрон. У нас есть тенденция использовать надежные правила при оптимизации.

Они являются:

* Количество фильтров: во время этого процесса размер карты объектов уменьшается с глубиной, таким образом, слои, близкие к входному слою, могут иметь тенденцию содержать меньше фильтров, в то время как более высокие слои будут иметь дополнительные. Защитные дополнительные данные, касающиеся входных данных, потребуются для того, чтобы общее разнообразие активаций не уменьшалось от одного уровня к следующему.
* Форма фильтра: при этом форма фильтра зависит от набора данных. Мы хотим найти правильный уровень, чтобы найти форму фильтра без каких-либо
* Максимальная форма объединения: при этом выбор больших фигур приводит к уменьшению размера сигнала, и это приведет к избытку данных

**Разные Слои**

Сверточная нейронная сеть, если она сформирована с совершенно разными слоями, которые преобразуют входной слой в выходной уровень ассоциированной степени. Слои нейронной сети упомянуты ниже:

* Сверточный слои
* Объединяющий слои
* Полносвязанный слой
* Слой классификации

**Регуляризация**

Сверточные нейронные сети используют различные формы регуляризации. Они являются:

* Эмпирическая регуляризация
* Явная Регуляризация

**Эмпирическая регулиризация**

При эмпирической регуляризации у нас есть тенденция иметь:

* Отсев: Отсев — это один из самых эффективных методов регуляризации, появившихся за последние пару лет. Основной план, лежащий в основе отсева, состоит в том, чтобы запускать каждую итерацию формулы декораций на случайно измененных версиях первого DLN.
* Drop connect: Это обобщение отсева. Drop Connect сравним с drop out, поскольку он вводит активную скудость внутри модели. В этом drop connect он работает так же, как и в dropout, но разница в том, что мы используем узлы вместо весов.
* Стохастическое объединение: при случайном объединении стандартные согласованные операции объединения заменяются случайной процедурой, при которой активация в каждой области объединения выбирается случайным образом в соответствии с многочленным распределением, заданным действиями в области объединения. Этот подход свободен от гиперпараметров и будет сочетаться с различными подходами к регуляризации, такими как отсев и увеличение информации. При случайном объединении у нас есть тенденция выбирать объединенный отклик карты путем выборки из многочленного распределения, сформированного из активаций каждой области объединения.
* Искусственные данные: поскольку степень переобучения модели определяется каждой ее мощностью и, следовательно, количеством получаемого ею обучения, предоставление сверточной сети дополнительных примеров обучения сократит переобучение. Эти сети, как правило, обучаются со всеми доступными знаниями, один из подходов заключается в том, чтобы либо генерировать новые знания с нуля, либо создавать новые.

**Экспресс регулиризация**

При экспресс-регуляризации у нас есть тенденция иметь:

* Ранняя остановка: Ранняя остановка заключается в том, что мысль может предотвратить переобучение. В этом случае набор информации используется для подсчета потерь в начале каждой эпохи коучинга, и как только потери перестают уменьшаться, остановите коучинг и используйте контрольные знания для расчета максимальной точности классификации.
* Количество параметров: В CNN размер фильтра дополнительно влияет на количество параметров. Ограничение количества параметров напрямую ограничивает прогностическую мощность сети, снижает качество, с которым они будут работать, что будет зависеть от информации, и, следовательно, ограничивает количество переобучений.
* Уменьшение веса: Уменьшение веса — это более простой метод регуляризации, который просто добавляет дополнительную ошибку, пропорциональную сумме весов или квадрату величины вектора нагрузки, к ошибке в каждом узле.
* Ограничения максимальной нормы: Регуляризация заключается в том, чтобы установить абсолютную границу степени для величины вектора нагрузки для каждой вегетативной клетки и использовать прогнозируемый градиентный спуск для обеспечения соблюдения ограничения.

**Приложения сверточной нейронной сети**

Сверточные нейронные сети используются во многих приложениях. Некоторые из них упоминаются ниже:

* Он используется для распознавания изображений
* Он используется для анализа видео
* Используется для языкового процесса
* Открытие лекарств
* Оценка Риска для здоровья
* Игра в шашки
* Прогнозирование временных рядов

## 2.2. Анализ и выбор прототипов НС

На основании вышеизложенного можно сделать вывод, что наиболее подходящей архитектурой НС для достижения цели проектирования является использование модели CNN. Как правило, глубокое обучение — это использование нейронных систем с избытком одиночного скрытого слоя нейронов. Тем не менее, это чрезвычайно упрощенная перспектива глубокого обучения, и она не является последовательной. Эти "глубокие" структуры дополнительно сильно колеблются, при этом различные исполнения улучшаются для различных заданий или целей. Огромные исследования, проводимые с такой постоянной скоростью, раскрывают новые и изобретательные модели глубокого обучения с постоянно растущими темпами [1].

### 2.2.1. Базовое ядро сети CNN

При классификации изображений мы предполагаем, что входное изображение содержит один объект, а затем мы должны классифицировать изображение в один из предварительно выбранных целевых классов с помощью моделей CNN. Некоторые из основных архитектур (моделей) CNN, предназначенных для классификации изображений, кратко описаны следующим образом:

**LeNet-5**

LeNet-5 - одна из самых ранних архитектур CNN, которая была разработана для классификации рукописных цифр. Он был представлен Леунгом и др. в 1998 году. LeNet-5 имеет 5 взвешенных (обучаемых) слоев, т. е. три сверточных слоя и два слоя FC. Среди них за каждым из первых двух слоев свертки следует слой с максимальным объединением (для карт объектов выборки), а затем за последним слоем свертки следуют два полностью соединенных слоя. Последний слой этих полностью соединенных слоев используется в качестве классификатора, который может классифицировать 10 цифр. Архитектура LeNet-5 показана на Рисунке 2.8.

Лейтмотив:

* LeNet-5 прошел обучение на цифровом наборе данных MNIST.
* В качестве функции активации он использовал нелинейность сигмоиды.
* Он использовал алгоритм обучения стохастическому градиентному спуску (SGD) с 20 периодами обучения.
* Он использовал 0,02 в качестве значения коэффициента импульса.
* Это снизило частоту ошибок тестирования до 0,95% в наборе данных MNIST.

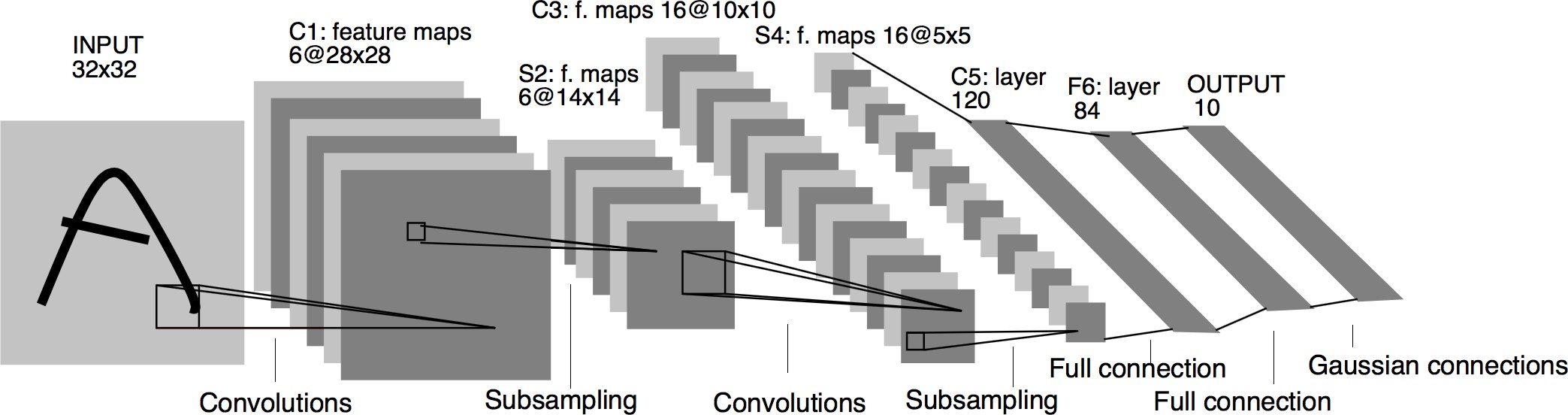
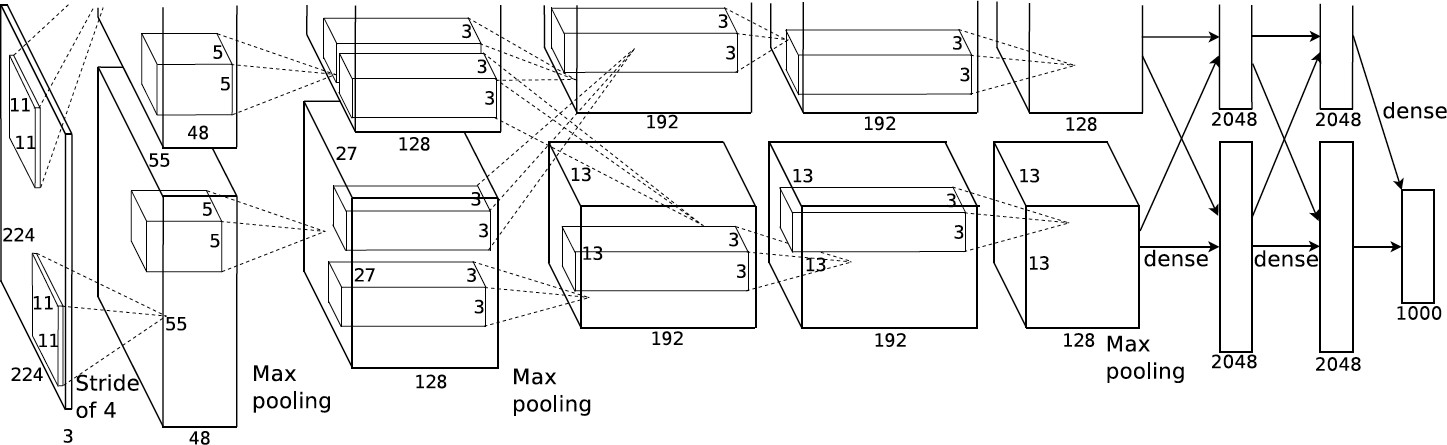


Рисунок 2.8. архитектура LeNet-5

**AlexNet**

Вдохновленный LeNet, Крижевский и др. разработана первая крупномасштабная модель CNN, получившая название AlexNet в 2012 году, которая предназначена для классификации данных ImageNet. Он состоит из восьми взвешенных (обучаемых) слоев, среди которых первые пять слоев являются сверточными слоями, а затем последние три слоя являются полностью связанными слоями. Поскольку он был разработан для данных ImageNet, поэтому последний выходной слой классифицирует входные изображения в один из тысячи классов набора данных ImageNet с помощью 1000 единиц [12]. Архитектура AlexNet показана на Рисунке 2.9.

Рисунок 2.9. Архитектура AlexNet

лейтмотив:

* AlexNet использовал функцию активации нелинейности выпрямленной линейной единицы (ReLU) после каждого сверточного и полностью связанного слоя.
* Он использовал слой максимального объединения после каждого слоя LRN и последнего сверточного слоя.
* Поскольку он имеет большее количество весов (обучаемых), поэтому, чтобы избежать переобучения, он использует несколько приемов регуляризации, таких как отсев и увеличение данных.
* Сеть AlexNet была обучена с использованием алгоритма обучения стохастического градиентного спуска (SGD) с минимальным размером пакета 128, снижением веса 0,0005 и значением коэффициента импульса 0,9.
* AlexNet был обучен (в наборе данных ImageNet) на двух NVIDIA GTX 580 (с 3 ГБ памяти) с использованием распараллеливания между графическими процессорами, и для его завершения требуется около шести дней.
* AlexNet стал победителем ILSVRC-2012.

**ZFNet**

ZFNet был представлен Цайлером и Фергусом в ECCV-2014, он имеет почти такую же архитектуру, как AlexNet, за исключением того, что здесь они использовали фильтр 7x7 с шагом 2 в 1-м сверточном слое. В случае AlexNet, Крижевки и др. используйте фильтр 11x11 с шагом 4 в 1-м сверточном слое. В результате ZFNet становится более эффективным, чем AlexNet, и становится победителем ILSVRC-2013. Архитектура Zdnet показана на Рисунке2.9.

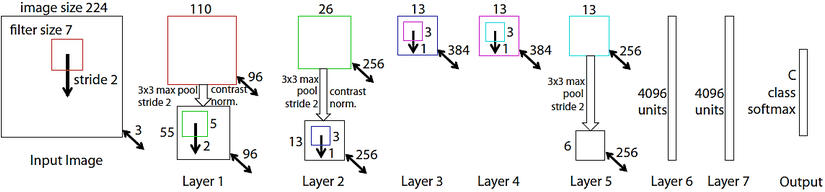


Рисунок 2.9. Архитектура Zdnet

**VGGNet**

VGGNet - одна из самых популярных архитектур CNN, представленная Симоняном и Зиссерманом в 2014 году. Авторы представили в общей сложности 6 различных конфигураций CNN, среди которых наиболее успешными являются VGGNet-16 (конфигурация D) и VGGNet-19 (конфигурация E)[11]. Архитектура VGGNet-16 показана на Рисунке 2.10.



Рисунок 2.10. Архитектура VGGNet-16

Лейтмотив:

* Причиной популярности VGGNet является его архитектурная простота и использование малогабаритных фильтров для сверточной работы.
* Это показывает, что стопка фильтров размером 3x3 имеет такое же эффективное приемное поле, как и фильтры большего размера при свертке (например, два слоя фильтров размером 3x3 имеют такое же эффективное приемное поле, как фильтры 5x5 при свертке, фильтры 7x7 с тремя слоями
* фильтров размером 3x3 и так далее). Самое главное, что использование малогабаритных фильтров уменьшает количество параметров сети.
* Сеть VGGNet была разработана и обучена на базе данных ILSVRC.
* Это очень глубокая сеть по сравнению с AlexNet.
* Он также использовал функцию активации нелинейности выпрямленной линейной единицы (ReLU) после каждого сверточного и полностью связанного слоя.
* Он также использует несколько приемов регуляризации, таких как отсев и увеличение данных, чтобы избежать чрезмерной подгонки.

**GoogLeNet**

Архитектура GoogLeNet отличается от всех ранее обсуждавшихся обычных моделей CNN, она использует сетевые ветви вместо использования однострочной последовательной архитектуры. Сеть GoogLeNet была предложена Сегеди и др. в 2014 году. В GoogLeNet есть 22 взвешенных (обучаемых) слоя, он использовал “Начальный модуль” в качестве основного строительного блока сети. Обработка этого модуля происходит параллельно в сети, и каждый (простой базовый) модуль состоит из отфильтрованных слоев свертки 1x1, 3x3 и 5x5 параллельно, а затем он объединяет их выходные карты объектов, что может привести к очень высокомерному выводу объектов. Для решения этой проблемы они использовали начальный модуль с уменьшением размерности (как показано на Рисунке 2.11(2)) в своей сетевой архитектуре вместо наивной (базовой) версии начального модуля (как показано на Рисунке 2.11(1)).

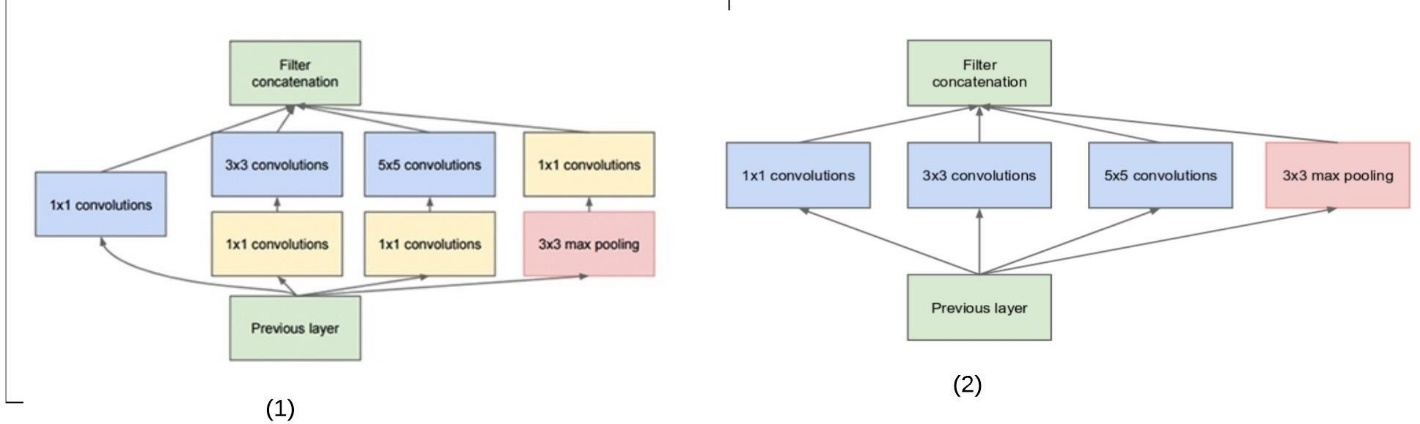


Рисунок 2.11. Архитектура GoogLeNet

Лейтмотив:

* Хотя GoogLeNet имеет 22 слоя, но у него в 12 раз меньше параметров, чем у AlexNet.
* Он имеет вспомогательные классификаторы, которые используются для борьбы с проблемой исчезающего градиента.
* Он также использовал функцию активации нелинейности выпрямленной линейной единицы (ReLU).
* Он использовал средний слой объединения вместо полностью подключенных слоев.
* В GoogLeNet использовался алгоритм обучения SGD с фиксированной скоростью обучения и коэффициентом импульса 0,9.
* GoogLeNet стал победителем ILSVRC-2014.

**ResNet**

Поскольку глубокая модель CNN страдает от проблем с исчезающим градиентом, как мы обсуждали ранее, он и др. из Microsoft представили идею “соединения с пропуском идентификации” для решения проблемы исчезающего градиента, предложив модель ResNet. Архитектура сети ResNet использует отображение остатков (H(x) = F(x) + x) вместо изучения прямого отображения (H(x) = F(x)), и эти болки называются остаточными блоками. Полная архитектура ResNet состоит из множества остаточных блоков со слоями свертки 3x3. На Рисунке 2.12 показана разница между прямым отображением и отображением остатков.

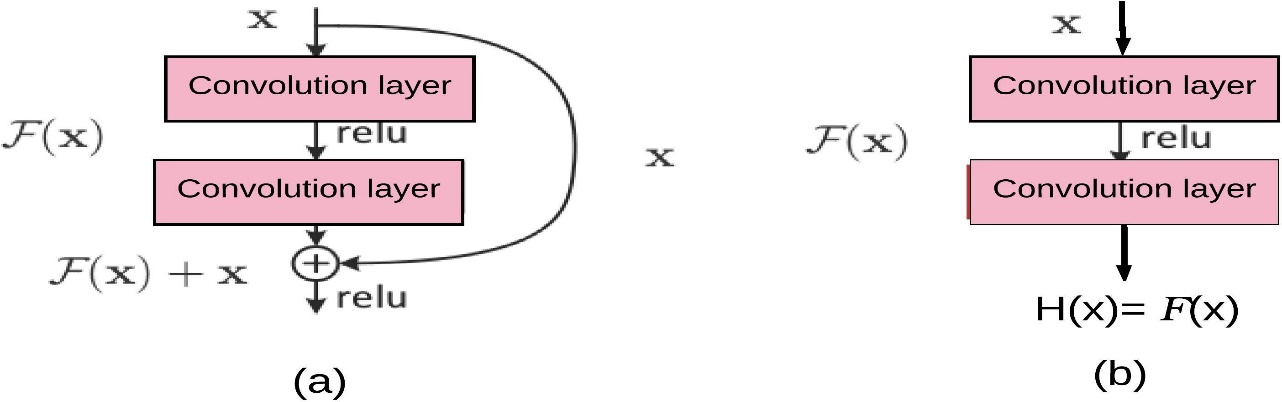


Рисунок 2.12 Разница между прямым отображением и отображением остатков

Лейтмотив:

* Авторы предлагают несколько версий ResNet с разной глубиной, а также использовали слой "узкого места" для уменьшения размерности в каждой архитектуре ResNet, глубина которой превышает 50.
* Хотя сеть ResNet (со 152 слоями) в 8 раз глубже, чем VGGNets (22 слоя), ее сложность ниже, чем у VGGNets (16/19).
* В сети ResNet использовался алгоритм обучения SGD с минимальным размером пакета 128, снижением веса 0,0001 и коэффициентом импульса 0,9.
* Сеть ResNet стала победителем ILSVRC-2015 с большим скачком производительности, она снижает частоту ошибок в топ-5 до 3,6% (у победителя предыдущего года GoogLeNet частота ошибок в топ-5 составляет 6,7%).

**DenseNet**

DenseNet расширяет идею отображения остатков, распространяя выходные данные каждого блока на все последующие блоки внутри каждого плотного блока в сети, как показано на Рисунке 2.13. Распространяя информацию как в прямом, так и в обратном направлениях во время обучения модели, она усиливает способность к распространению признаков и решает проблему исчезающего градиента. Плотная сеть была представлена Хуангом и др. в 2016 году и стала победителем ILSVRC-2016. Инжир. показывает модель, основанную на плотной сети.

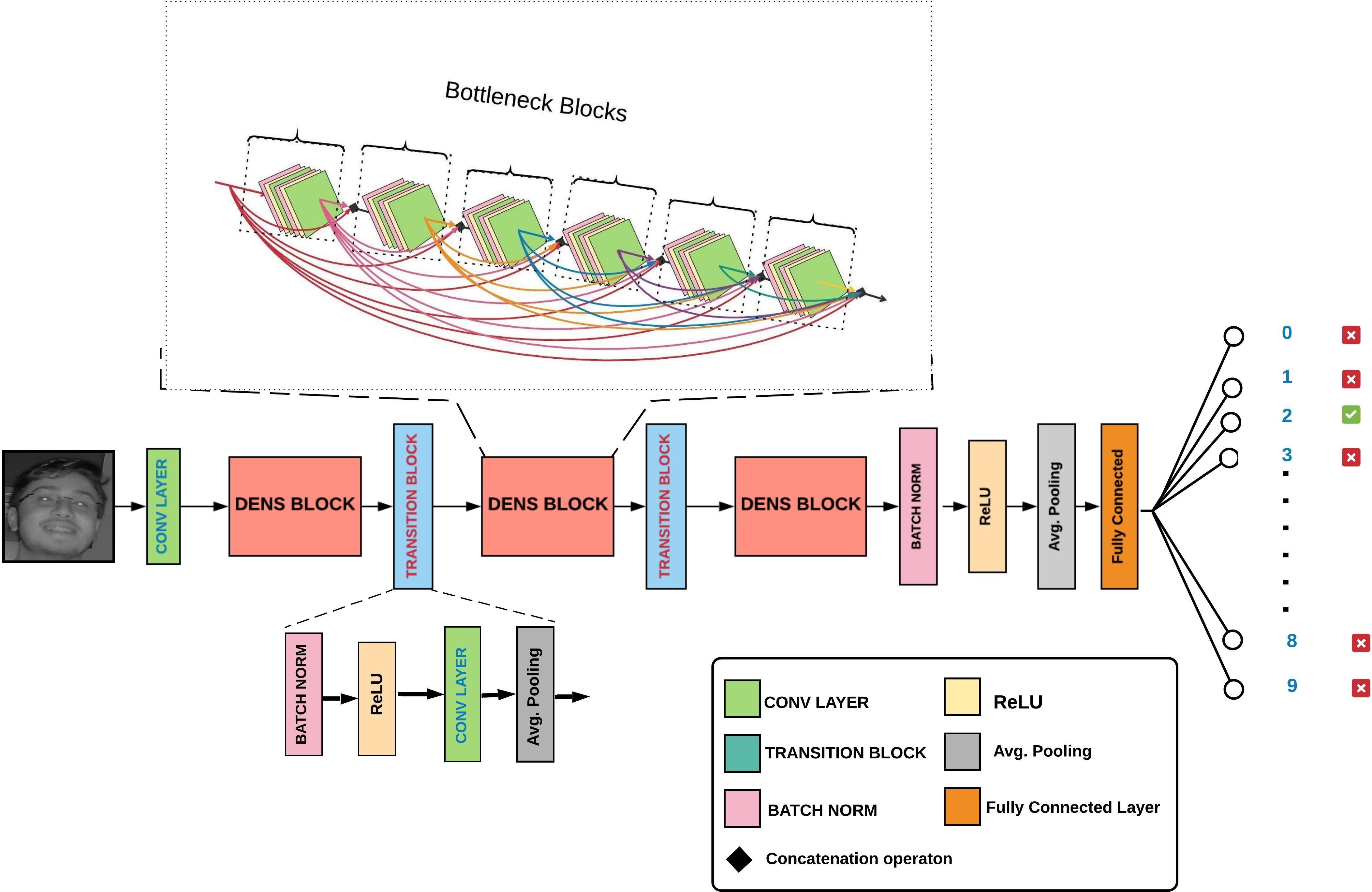


Рисунок.2.13 Архитектура DenseNet

Рассмотрим одно изображение x0, которое передается через сверточную сеть. Сеть состоит из L слоев, каждый из которых реализует нелинейное преобразование Hl(·), где l индексирует слой. Hl(·) может быть составной функцией таких операций, как нормализация партии (BN), выпрямленные линейные единицы (ReLU), объединение или Свертка (Conv). Мы обозначаем вывод l го слоя как xl.

**Выводы**

На основании вышеизложенного можно сделать вывод что наиболее рациональным подходом является использование в качестве базовой архитектуры сети архитектуру нейронной сети VGG16/19 для достижения цели проектирования

## 2.3. Анализ и выбор методов обучения НС

Обучение нейронной сети — это процесс, в котором параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть встроена. Тип обучения определяется способом подстройки параметров. Различают алгоритмы обучения с учителем и без учителя. Процесс обучения с учителем представляет собой предъявление сети выборки обучающих примеров. Каждый образец подается на входы сети, затем проходит обработку внутри структуры НС, вычисляется выходной сигнал сети, который сравнивается с соответствующим значением целевого вектора, представляющего собой требуемый выход сети.

Для того, чтобы нейронная сети была способна выполнить поставленную задачу, ее необходимо обучить. Различают алгоритмы обучения с учителем и без учителя. Процесс обучения с учителем представляет собой предъявление сети выборки обучающих примеров. Каждый образец подается на входы сети, затем проходит обработку внутри структуры НС, вычисляется выходной сигнал сети, который сравнивается с соответствующим значением целевого вектора, представляющего собой требуемый выход сети. Затем по определенному правилу вычисляется ошибка, и происходит изменение весовых коэффициентов связей внутри сети в зависимости от выбранного алгоритма. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

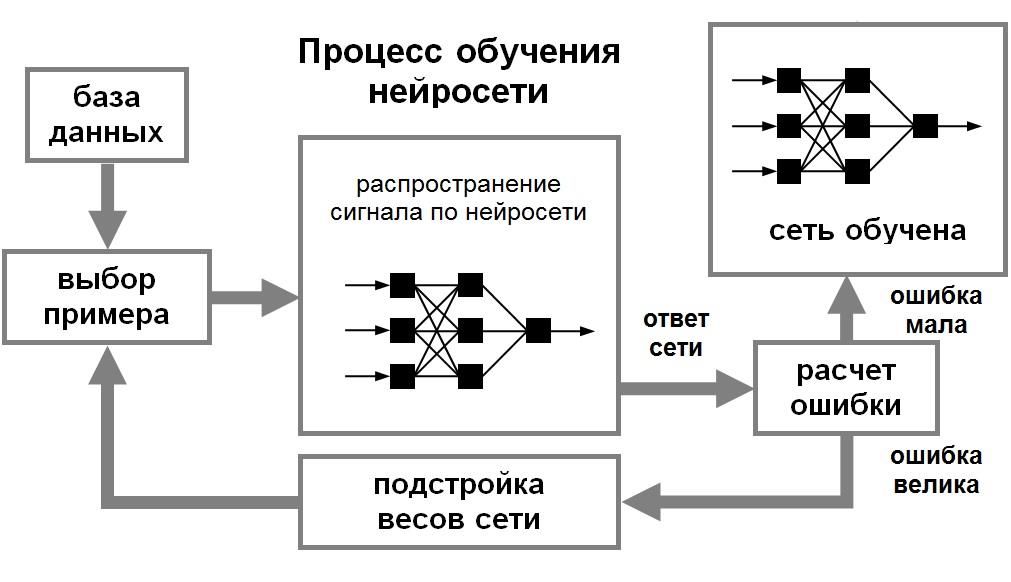


Рисунок 2.14. Процесс обучения нейросети

При обучении без учителя обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса сети так, чтобы получались согласованные выходные векторы, т.е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет статистические свойства обучающего множества и группирует сходные векторы в классы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, но до обучения невозможно предсказать, какой выход будет производиться данным классом входных векторов. Следовательно, выходы подобной сети должны трансформироваться в некоторую понятную форму, обусловленную процессом обучения. Это не является серьезной проблемой. Обычно не сложно идентифицировать связь между входом и выходом, установленную сетью. Для обучения нейронных сетей без учителя применяются сигнальные метод обучения Хебба и Ойа.

Математически процесс обучения можно описать следующим образом. В процессе функционирования нейронная сеть формирует выходной сигнал Y, реализуя некоторую функцию Y = G(X). Если архитектура сети задана, то вид функции G определяется значениями синаптических весов и смещенной сети.

Пусть решением некоторой задачи является функция Y = F(X), заданная параметрами входных-выходных данных (X1, Y1), (X2, Y2), …, (XN, YN), для которых Yk = F(Xk) (k = 1, 2, …, N).

Обучение состоит в поиске (синтезе) функции G, близкой к F в смысле некоторой функции ошибки E.

Если выбрано множество обучающих примеров – пар (XN, YN) (где k = 1, 2, …, N) и способ вычисления функции ошибки E, то обучение нейронной сети превращается в задачу многомерной оптимизации, имеющую очень большую размерность, при этом, поскольку функция E может иметь произвольный вид обучение в общем случае – многоэкстремальная невыпуклая задача оптимизации.

Для решения этой задачи могут использоваться следующие (итерационные) алгоритмы:

1. алгоритмы локальной оптимизации с вычислением частных производных первого порядка:

* градиентный алгоритм (метод наискорейшего спуска),
* методы с одномерной и двумерной оптимизацией целевой функции в направлении антиградиента,
* метод сопряженных градиентов,
* методы, учитывающие направление антиградиента на нескольких шагах алгоритма;

2. алгоритмы локальной оптимизации с вычислением частных производных первого и второго порядка:

* метод Ньютона,
* методы оптимизации с разреженными матрицами Гессе,
* квазиньютоновские методы,
* метод Гаусса-Ньютона,
* метод Левенберга-Марквардта и др.;

3. стохастические алгоритмы оптимизации:

* поиск в случайном направлении,
* имитация отжига,
* метод Монте-Карло (численный метод статистических испытаний);

4. алгоритмы глобальной оптимизации (задачи глобальной оптимизации решаются с помощью перебора значений переменных, от которых зависит целевая функция).

## 2.4. Анализ и выбор методов предварительного анализа, очистки, обработки и разбиения исходных данных

Предварительная обработка данных включает в себя шаги, которые нам необходимо выполнить для преобразования или кодирования данных, чтобы они могли быть легко проанализированы машиной. Основная задача для того, чтобы модель была точной и точной в прогнозах, заключается в том, чтобы алгоритм мог легко интерпретировать характеристики данных.

Большинство реальных наборов данных очень чувствительны к отсутствующим, противоречивым и зашумленным данным из-за их разнородного происхождения. Применение алгоритмов интеллектуального анализа данных к этим зашумленным данным не даст качественных результатов, поскольку они не смогут эффективно идентифицировать закономерности. Поэтому обработка данных важна для улучшения общего качества данных.

Существует четыре основных этапа обработки данных. Они являются

* Очистка данных,
* Интеграция
* данных, Преобразование данных и
* Сокращение данных или Уменьшение размеров.

Очистка данных

Очистка данных, в частности, выполняется как часть предварительной обработки данных для очистки данных путем заполнения пропущенных значений, сглаживания зашумленных данных, устранения несоответствий и удаления выбросов.

1. Пропущенные значения

Вот несколько способов решить эту проблему:

Игнорируйте эти кортежи:Этот метод следует рассматривать, когда набор данных огромен и в кортеже пРисутствуют многочисленные пропущенные значения.

Заполните недостающие значения:существует множество методов для достижения этой цели, таких как заполнение значений вручную, прогнозирование недостающих значений с помощью метода регрессии или численных методов, таких как среднее значение атрибута.

2. Зашумленные Данные

Это включает в себя устранение случайной ошибки или отклонения в измеряемой переменной. Это можно сделать с помощью следующих приемов:

Биннинг:Это метод, который работает с отсортированными значениями данных, чтобы сгладить любой пРисутствующий в них шум. Данные разделяются на ячейки одинакового размера, и каждая ячейка /корзина обрабатывается независимо. Все данные в сегменте могут быть заменены его средними, медианными или граничными значениями.

Регрессия:Этот метод интеллектуального анализа данных обычно используется для прогнозирования. Это помогает сгладить шум, подгоняя все точки данных в регрессионную функцию. Уравнение линейной регрессии используется, если имеется только один независимый атрибут; в противном случае используются полиномиальные уравнения.

Кластеризация:Создание групп/кластеров из данных, имеющих схожие значения. Значения, которые не находятся в кластере, могут рассматриваться как зашумленные данные и могут быть удалены.

3. Удаление выбросов

Методы кластеризации группируют похожие точки данных. Кортежи, которые находятся за пределами кластера, являются выбросами / несогласованными данными.

Интеграция данных

Интеграция данных — это один из этапов предварительной обработки данных, который используется для объединения данных, имеющихся в нескольких источниках, в единое более крупное хранилище данных, например хранилище данных.

Интеграция данных необходима, особенно когда мы стремимся решить реальный сценарий, такой как обнаружение наличия узелков на изображениях компьютерной томографии. Единственный вариант - объединить изображения с нескольких медицинских узлов для формирования более крупной базы данных.

Мы можем столкнуться с некоторыми проблемами при внедрении интеграции данных в качестве одного из этапов предварительной обработки данных:

* Интеграция схемы и сопоставление объектов: Данные могут быть представлены в разных форматах и с атрибутами, которые могут вызвать трудности при интеграции данных.
* Удаление избыточных атрибутов из всех источников данных.
* Обнаружение и разрешение конфликтов значений данных.

Преобразование данных

Как только очистка данных будет выполнена, нам необходимо объединить качественные данные в альтернативные формы, изменив значение, структуру или формат данных, используя перечисленные ниже стратегии преобразования данных.

Обобщение

Низкоуровневые или детализированные данные, которые мы преобразовали в высокоуровневую информацию с помощью концептуальных иерархий. Мы можем преобразовать примитивные данные в адресе, такие как город, в информацию более высокого уровня, такую как страна.

Нормализация

Это наиболее важный широко используемый метод преобразования данных. Числовые атрибуты масштабируются вверх или вниз, чтобы соответствовать заданному диапазону. В этом подходе мы ограничиваем наш атрибут данных определенным контейнером, чтобы создать корреляцию между различными точками данных. Нормализация может быть выполнена несколькими способами, которые выделены здесь:

* Нормализация Min-max
* Нормализация Z-балла
* Нормализация десятичного масштабирования

Выбор атрибута

Новые свойства данных создаются на основе существующих атрибутов, чтобы помочь в процессе интеллектуального анализа данных. Например, дата рождения, атрибут данных может быть преобразован в другое свойство, такое как is\_senior\_citizen для каждого кортежа, что напрямую повлияет на прогнозирование заболеваний или шансов на выживание и т.д.

Агрегация

Это метод хранения и представления данных в сводном формате. Например, данные о продажах могут быть объединены и преобразованы для отображения в формате месяца и года.

Сокращение объема данных

Размер набора данных в хранилище данных может быть слишком большим, чтобы его можно было обработать с помощью алгоритмов анализа данных и интеллектуального анализа данных.

Одним из возможных решений является получение уменьшенного представления набора данных, который намного меньше по объему, но дает такое же качество аналитических результатов.

Вот пошаговое руководство по различным стратегиям сокращения объема данных.

Агрегация кубов данных

Это способ сокращения данных, при котором собранные данные выражаются в сводной форме.

Уменьшение размерности

Для извлечения объектов используются методы уменьшения размерности. Размерность набора данных относится к атрибутам или отдельным характеристикам данных. Этот метод направлен на уменьшение количества избыточных функций, которые мы рассматриваем в алгоритмах машинного обучения. Уменьшение размерности может быть выполнено с использованием таких методов, как анализ основных компонентов и т.д.

Сжатие данных

При использовании технологий кодирования размер данных может значительно сократиться. Но сжатие данных может быть как с потерями, так и без потерь. Если исходные данные могут быть получены после восстановления из сжатых данных, это называется уменьшением без потерь; в противном случае это называется уменьшением с потерями.

Дискретизация

Дискретизация данных используется для разделения атрибутов непрерывного характера на данные с интервалами. Это делается потому, что непрерывные объекты, как правило, имеют меньшую вероятность корреляции с целевой переменной. Таким образом, интерпретировать результаты может быть сложнее. После дискретизации переменной могут быть интерпретированы группы, соответствующие цели. Например, возраст атрибута может быть дискретизирован в ячейки типа «ниже 18», «18-44», «44-60», «выше 60».

Уменьшение численности

Данные могут быть представлены в виде модели или уравнения, подобного регрессионной модели. Это избавило бы от необходимости хранить огромные наборы данных вместо модели.

Выбор подмножества атрибутов

Очень важно быть конкретным в выборе атрибутов. В противном случае это может привести к получению данных с высокой размерностью, которые трудно обучить из-за проблем с недостаточной / чрезмерной подготовкой. Следует учитывать только те атрибуты, которые повышают ценность обучения модели, а все остальное можно отбросить.

**Выводы**

На основании вышеизложенного можносделать вывод, что наиболее рациональным подходом явялется использование метода обучения с учителем.

## 2.5. Анализ и выбор среды для разработки НС

Фреймворки глубокого обучения, такие как PyTorch, Tensorflow, Karas и Cafe, очень эффективны для того, чтобы заставить машины учиться, как люди, с помощью специальных мозгоподобных архитектур, известных как нейронные сети. Теперь я расскажу об этих фреймворках. Без правильной структуры построение качественных нейронных сетей может быть затруднено. При правильной структуре вам нужно беспокоиться только о том, чтобы получить в свои руки нужные данные. Это не означает, что одного знания фреймворков глубокого обучения достаточно, чтобы сделать вас успешным специалистом по обработке данных. Кроме того, не все языки программирования имеют свои собственные фреймворки машинного обучения / глубокого обучения. Это связано с тем, что не все языки программирования способны решать проблемы машинного обучения. Такие языки, как Python, выделяются среди других благодаря своим сложным возможностям обработки данных.

### 2.5.1. PyTorch

PyTorch – это библиотека с открытым исходным кодом, используемая в библиотеке машинного обучения, разработанная с использованием библиотеки Torch для программы python. Он разработан исследовательской лабораторией искусственного интеллекта Facebook и выпущен в январе 2016 года как бесплатная библиотека с открытым исходным кодом, которая в основном используется в приложениях компьютерного зрения, глубокого обучения и обработки естественного языка. Программист может с легкостью построить сложную нейронную сеть, используя Pitch, поскольку она имеет основную структуру данных, тензор, многомерный массив, подобный массивам Numpy. Использование PyTorch растет в современных отраслях промышленности и в исследовательском сообществе, поскольку он является гибким, быстрым и простым в запуске проекта, благодаря чему PyTorch является одним из лучших инструментов глубокого обучения.

**Почему нам нужен PyTorch?**

Фреймворк PyTorch можно рассматривать как будущее фреймворка глубокого обучения. Внедряется множество фреймворков глубокого обучения, и наиболее предпочтительными фреймворками являются Tensorflow и PyTorch, но среди всех PyTorch становится победителем благодаря своей гибкости и вычислительной мощности. Для энтузиастов машинного обучения и искусственного интеллекта PyTorch прост в освоении и будет очень полезен для построения моделей.

Вот некоторые из причин, по которым разработчики и исследователи изучают PyTorch:

1. Легко усваивается

PyTorch имеет ту же структуру, что и традиционное программирование, и она была блестяще задокументирована сообществом разработчиков, постоянно работающим над улучшением. Благодаря этому его легко освоить как программисту, так и непрограммисту.

2. Производительность разработчиков

Он имеет интерфейс с python и различными мощными API-интерфейсами и может быть реализован в ОС Windows или Linux. С помощью некоторого программирования разработчик знаний может повысить свою производительность, поскольку большинство задач из PyTorch могут быть автоматизированы.

3. Простота отладки

Он может использовать инструменты отладки, такие как pdb и pdb tools python. Поскольку PyTorch разрабатывает вычислительный граф во время выполнения, программист может использовать Pythons IDE PyCharm для отладки.

4. Параллелизм данных

Он может распределять вычислительные задачи между несколькими процессорами или графическими процессорами. Это возможно с помощью функции параллелизма данных (torch.nn.Data Parallel), которая обертывает любой модуль и помогает нам выполнять параллельную обработку.

5. Полезные Библиотеки

В нем есть большое сообщество разработчиков и исследователей, которые создали инструменты и библиотеки для расширения PyTorch. Это сообщество помогает в развитии компьютерного зрения, обучении с подкреплением, НЛП для исследовательских и производственных целей. некоторые из популярных библиотек - GPyTorch, Botorch и Allen NLP. Богатый набор мощных API-интерфейсов помогает расширить фреймворк Python.

**Компоненты PyTorch**

Давайте рассмотрим пять основных компонентов PyTorch:

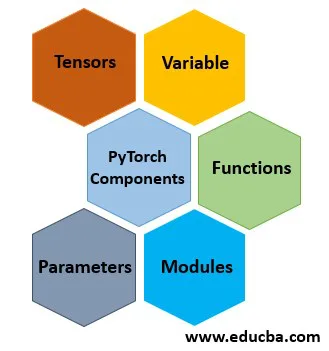


Рисунок 2.15. Компоненты PyTorch

1. Тензоры: Тензоры — это многомерный массив, аналогичный массиву Numpy, и тензоры доступны в Torch как torch. Интенсор, факел. Плавающий стабилизатор, факел.Чартен и т.д.

2. Переменная: Переменная — это, по сути, оболочка вокруг тензоров для удержания градиента. Переменная доступна в разделе torch.autograd как torch.autograd.Variable.

3. Параметры: Параметры — это, по сути, оболочка вокруг переменной. Он используется, когда мы хотим, чтобы тензор был параметром некоторого модуля, что невозможно с помощью переменной, поскольку это не параметр или тензоры, у которых нет градиента, поэтому мы можем использовать параметры в torch.nn в качестве torch.nn.Parameter.

4. Функции: Функции выполняют операции преобразования, и у них нет памяти для хранения какого-либо состояния. Подобные логарифмические функции будут выдавать выходные данные в виде логарифмического значения, а линейный слой не может функционировать, поскольку он сохраняет вес и смещает значение. Примерами функций являются torch.log, torch.sum и т.д., А функции реализованы в рамках torch.nn.functional.

**5. Модули:** Модуль, используемый как под Torch в качестве torch.nn.Module, является базовым классом для всех нейронных сетей. Модуль может содержать другие модули, параметры и функции. Он может хранить состояние и обучаемые веса. Модуль - это типы преобразований, которые могут быть реализованы как torch.nn.Conv2d, torch.nn.Linear и т.д.

**Преимущества**

* Он прост в освоении и проще в программировании.
* Богатый набор мощных API-интерфейсов для расширения библиотек Pytorch.
* Он имеет поддержку вычислительных графиков во время выполнения.
* Он является гибким, быстрым и обеспечивает оптимизацию.
* Он имеет поддержку графического процессора и центрального процессора.
* Легко отлаживается с помощью Pythons IDE и инструментов отладки.
* Он поддерживает облачные платформы.

**Недостатки**

* Меньшее сообщество, чем у TensorFlow.
* Нет такого инструмента визуализации, как Tensorboard
* Меньше инструментов для производственных развертываний (например, Tensorflow Lite)

**Применение PyTorch**

**1. Компьютерное зрение**

В нем используется сверточная нейронная сеть для разработки классификации изображений, обнаружения объектов и генеративного приложения. Используя PyTorch, программист может обрабатывать изображения и видео для разработки высокоточной и точной модели компьютерного зрения.

**2. Обработка естественного языка**

Его можно использовать для разработки языкового переводчика, языкового моделирования и для разработки чат-бота. Он использует RNN, LSTM и т.д. Архитектура для разработки естественного языка, моделей обработки.

**3. Обучение с подкреплением**

Он используется для разработки робототехники для автоматизации, планирования бизнес-стратегии или управления движением робота и т.д. Он использует архитектуру глубокого Q-обучения для построения модели.

**Вывод**

Это один из основных фреймворков, используемых для глубокого обучения, обработки естественного языка и т.д. В будущем все больше и больше исследователей и разработчиков будут изучать и внедрять PyTorch. Он имеет такой же синтаксис, как и любой другой стандартный язык программирования; следовательно, он предназначен для изучения и перехода в области искусственного интеллекта или инженера машинного обучения.

### 2.5.2. Tensorflow

Он определяется как основа для шаблонов и устройств. Это дружественный python с открытым исходным кодом и библиотекой символьной математики, предназначенный для создания и проектирования моделей глубокого обучения с использованием графиков потоков данных. И выпущена Google как библиотека машинного обучения с открытым исходным кодом. Библиотека TensorFlow выполняет многочисленные вычисления с помощью графиков потоков данных.

Машинное обучение — это быстро развивающаяся технология в сфере бизнеса; несколько секторов используют их для крупных предприятий. Правильное использование этой технологии имеет большое значение; чтобы спасти это, Google разработал TensorFlow и сделал его открытым исходным кодом в 2015 году. У них есть множество встроенных функций и обработки данных; это проще при разработке нового алгоритма. С другой стороны, он предоставляет полную инфраструктуру для работы с машинным обучением; в основном он используется в исследовательских работах. Машинное обучение выявляет сложные закономерности в данных о системах для принятия правильных решений. Он создается, поскольку имеет ограниченную вычислительную мощность и используется для обслуживания прогнозов.

Он состоит из трех основных компонентов, это:

* API тензорного потока
* Тензорный поток, обслуживающий
* Тензорная доска

**Понимание**

Tensor является наиболее широко используемой платформой, поскольку ее гибкость также обеспечивает хорошее удобство для отладки в приложениях TensorFlow. Его можно рассматривать как хорошую систему программирования, в которой операции развертываются в виде графиков. Он выполняется на различных платформах, а установка выполняется с использованием среды pip. Тензор имеет несколько измерений данных, которые представлены с использованием ранга. Он предоставляет API-интерфейсы для работы с программами GO, где вы можете импортировать и определять графики. Узлы представляют математические операции; ребро представляет многомерный массив данных. Это приложение работает на локальном компьютере, устройствах Android, Google customs.

**Как TensorFlow делает работу такой простой?**

Это делает работу такой простой и удобной. Наиболее важной особенностью является тензорная плата, которая позволяет нам графически визуализировать и отслеживать работу тензора. Машинное обучение во многом зависит от матричных концепций, доступных в многомерном массиве; оно работает очень быстро при вычислении матриц и может быть доступно на таких языках, как Python, C++. Этот инструмент настолько гибок в работе благодаря своим библиотечным API, работающим на CPU и GPU. Вы можете загружать данные двумя лучшими способами: загружать данные в память, конвейер данных. Эти методы очень хорошо работают с большими наборами данных.

**Что вы можете сделать с TensorFlow?**

Хорошо известно, что создавать методы обучения, собирать данные, внедрять методы обучения, анализировать прогнозы и, наконец, получать будущие результаты. С помощью простой строки кода на python создается последовательная нейронная сеть. И далее, с помощью javascript, мы можем обучить образцы наборов данных и выполнить их в браузере, используя расширение .js. Многие варианты использования связаны с TensorFlow; популярными примерами являются текстовые приложения, такие как определение языка и сентиментальный анализ. Далее идет распознавание изображений, а также работа над распознаванием видео.

**Преимущества**

Ниже приведены упомянутые преимущества:

* Преимущество его использования заключается в том, что он обеспечивает абстракцию для реализации машинного обучения.
* Они эффективно работают со сложными математическими вычислениями с многомерными массивами.
* Прелесть этого в том, что у них лучшая визуализация графиков. Вы можете визуализировать каждое направление графика с помощью адаптивной конструкции. Самое лучшее, что они имеют открытый исходный код и легко настраиваются с помощью различных удивительных библиотечных продуктов и хорошо работают в распределенных вычислениях.
* Они предлагают конвейеру обучать несколько нейронных сетей параллельно.

**Почему мы должны использовать TensorFlow?**

Используя это, мы можем создавать хорошие визуализации и документацию и пользоваться широкой поддержкой сообщества. Он в основном вдохновлен, поскольку используется для классификации, обнаружения прогнозов, выявления закономерностей и применения восприятия и творчества. Он использовался в приложениях машинного обучения и производственной части Google для разработки оптимизированного решения. Такие приложения, как здравоохранение, продукты Google, социальные сети, реклама, используют передовое машинное обучение, а TensorFlow помогает достичь своей цели.

**Масштаб**

Программное обеспечение Tensorflow постоянно обновляется и в ближайшие годы будет стремительно развиваться. Это полностью считается будущим моделирования машинного обучения. Многие ведущие компании используют его для своих исследовательских аспектов, таких как Bloomberg, google, intel, deep mind, GE health care, eBay и т.д. Они наиболее известны тем, что находят свою роль в крупных компаниях, академических кругах, особенно в продуктах Google. Даже они начали свой рабочий путь в облаке, на мобильных устройствах.

Наличие графических моделей хорошо подходит для развертывания нейронных сетей. Вспомогательные библиотеки TensorFlow помогают в отладке, визуализируя реализованные им модели. Вы можете легко внедрить алгоритмы глубокого обучения, и это инновационная технология, создающая многочисленные возможности для карьерного роста.

**Как эта технология поможет вам в карьерном росте?**

По данным сообщества tensor, облачные технологии и большие данные продолжают стремительно развиваться на рынке, где они используют методы глубокого обучения. Понятно, что при обучении TensorFlow будет востребован специалист по глубокому обучению. У них есть лучший карьерный рост, поскольку они умнее справляются со сложными проблемами изучения данных. Он решает широкий спектр проблем искусственного интеллекта; следовательно, он открывает хорошие возможности для трудоустройства в среде анализа данных. Многие учебные заведения, ориентированные на профессиональную подготовку, занимаются этим обучением, чтобы кандидаты соответствовали требованиям отрасли.

Преимущества:

* Огромное; вероятно, самое большое сообщество разработчиков и исследователей ML.
* Тщательная документация и руководящие принципы.
* Визуализация с помощью TensorBoard упрощает проектирование и отладку модели.
* Tensorflow Lite позволяет развертывать на мобильных и периферийных устройствах.
* Tensorflow JS позволяет развертывать в средах JavaScript.
* Поддерживает распределенные вычисления.
* Обслуживание TensorFlow обеспечивает гибкую, высокопроизводительную систему обслуживания моделей машинного обучения, разработанную для производственных сред.
* Статический график вычислений отлично подходит для повышения производительности и обеспечивает возможность запуска на разных устройствах (CPU / GPU / TPU).

**Недостатки:**

* Высокий барьер для входа для начинающих.
* Сложный дизайн системы, на GitHub более 1 миллиона строк исходного кода, что затрудняет полное понимание фреймворка.
* Часто изменяемые API-интерфейсы. API Tensorflow быстро повторяется, и обратная совместимость не была должным образом рассмотрена. Это привело к тому, что многие проекты с открытым исходным кодом оказались несовместимыми с последней версией TensorFlow.
* Предоставляет множество реализаций для одной и той же функциональности, что затрудняет пользователям выбор.

### 2.5.3. Caffe

Caffe, популярная платформа глубокого обучения с открытым исходным кодом, была разработана Berkley AI Research. Он обладает высокой выразительностью, модульностью и быстродействием. Он имеет богатую документацию с открытым исходным кодом, доступную на Github. Он широко используется в академических исследовательских проектах, в доказательстве концепций стартапов, компьютерном зрении, обработке естественного языка и т.д.

**Платформа глубокого обучения Caffe**

Он расшифровывается как Сверточная архитектура для быстрого встраивания функций и написан в лицензированной BSD библиотеке C++ с привязками Python и MATLAB. Он используется для эффективного обучения и развертывания сверточных нейронных сетей общего назначения на товарных архитектурах. Архитектура вышеупомянутого фреймворка в целом делится на следующие:

1. Хранение данных

Он использует данные N-мерного массива в C-непрерывном виде, называемые большими двоичными объектами, для хранения и передачи данных. Большой двоичный объект можно рассматривать как уровень абстракции между процессором и графическим процессором. Данные из центрального процессора загружаются в большой двоичный объект, который затем передается в графический процессор для вычисления. Под капотом blob-объект использует класс SyncedMem для синхронизации значений между CPU и GPU. Затем большой двоичный объект перемещается на следующий уровень без учета более низких деталей реализации при сохранении высокого уровня производительности. Для эффективного использования памяти используется метод отложенного выделения памяти по требованию для хоста и устройства. Для крупномасштабных данных используются базы данных LevelDB. Модели глубокого обучения хранятся во вторичном хранилище в виде буферов протокола Google, которые обеспечивают эффективную сериализацию, удобочитаемый текстовый формат и т.д.

2. Слои

Большие двоичные объекты передаются ему в качестве входных данных и, соответственно, генерируются выходные данные. Это следует из отношения "многие ко многим". В рамках типовых операций на него возложены следующие ключевые обязанности:

Настройка: Он инициализирует слой и базовые соединения в первый раз во время инициализации модели.

Прямой проход: Входные данные передаются и, соответственно, генерируются выходные данные.

Обратный проход: вычисление градиентов относительно выходных данных, гиперпараметров модели и входных данных, которые затем передаются последующим слоям с использованием метода, называемого обратным распространением.

Он обеспечивает различные настройки уровней, такие как свертка, объединение, нелинейные активации, такие как выпрямленные, линейные единицы измерения (ReLU) с широко используемыми потерями при оптимизации, такими как логарифмические потери, R-квадрат, Softmax и т.д. Слои могут быть расширены до новой пользовательской реализации пользовательского уровня с использованием композиционного построения сетей.

3. Сети и Базовая модель запуска

Он использует структуру данных, называемую направленным ациклическим графом, для хранения операций, выполняемых нижележащими слоями, обеспечивая таким образом правильность прямого и обратного проходов. Типичная сеть модели Caffe начинается со уровня данных, загружающего данные с диска, и заканчивается уровнем потерь, основанным на требованиях приложения. Он может быть запущен на CPU / GPU, и переключение между ними происходит плавно и независимо от модели.

4. Обучение сети

Типичная модель Caffe обучается с помощью быстрого и стандартного алгоритма стохастического градиентного спуска. Данные могут быть обработаны в виде мини-пакетов, которые последовательно передаются по сети. Важные параметры, связанные с обучением, такие как графики снижения скорости обучения, импульс и контрольные точки для остановки и возобновления, хорошо реализованы с тщательной документацией. Он также поддерживает тонкую настройку, метод, при котором существующая модель может быть использована для поддержки новой архитектуры или данных. Веса предыдущей модели обновляются для нового приложения, и новые веса назначаются везде, где это необходимо. Этот метод широко используется во многих реальных приложениях глубокого обучения.

**Преимущества платформы глубокого обучения Caffe**

Он предоставляет полный набор пакетов для обучения, тестирования, точной настройки и развертывания модели. В нем приводится множество примеров вышеуказанных задач. Раньше он использовался для задач зрения, но теперь его пользователи используют для других приложений глубокого обучения, таких как распознавание речи, нейронные сети, робототехника. Он может быть запущен на облачных платформах с плавным переключением платформ.

Модульность: возможно расширение до новых данных, слоев и функций оптимизации потерь. В нем есть справочные примеры, в которых реализованы слои и функции потерь.

Скорость: Его можно использовать для обработки 60 миллионов изображений в день с использованием графического процессора NVIDIA CUDA K40. Это одна из самых быстрых реализаций convnet, доступных на рынке.

Многоуровневая архитектура и реализация: Определения модели написаны с использованием языка буфера протокола в виде файлов конфигурации. Сетевая архитектура использует подход направленного ациклического графа. Когда модель создается, она резервирует точную память в соответствии с требованиями модели. Переключение из среды, основанной на процессоре, в среду GPU требует одного вызова функции.

Тестовое покрытие: Каждый модуль в нем тестируется, и его проект с открытым исходным кодом не допускает фиксации какого-либо модуля без соответствующих тестов, что позволяет быстро улучшать и рефакторинг кодовой базы. Таким образом, это повышает его ремонтопригодность, относительно свободную от ошибок / дефектов.

Поддержка Python и MATLAB в многослойности: обеспечивает интерфейс и простоту использования с существующей исследовательской платформой, используемой научными учреждениями. Оба языка могут использоваться для построения сети и классификации входных данных. Python в Layering также позволяет использовать модуль solver для разработки новых методов обучения и упрощения использования.

Ранее обученные модели, используемые в качестве эталонных: Он предоставляет эталонные модели для исследований и научных проектов, таких как обученные модели landmark ImageNet и т.д. Таким образом, он предоставляет общий программный компонент, который может быть расширен для быстрого прогресса в разработке модельных архитектур для реальных приложений.

Он отличается от других современных фреймворков CNN следующим:

Реализация в основном основана на C++, поэтому ее легко интегрировать в существующие системы C++ и общие отраслевые интерфейсы. Кроме того, режим процессора устраняет необходимость в специализированной аппаратной платформе для развертывания модели и экспериментов после обучения модели.

Эталонные модели предоставляются в готовом виде для быстрого экспериментирования с современными результатами. Таким образом, это снижает затраты на повторное обучение.

**Преимущества**

* Быстрое начало работы: вы можете перейти на свой собственный набор данных, не написав много кода.
* Выдающаяся производительность и быстрое прототипирование.
* Быстрая тренировка.

**Недостатки**

* Для развертывания вам необходимо выполнить компиляцию из исходного кода, а поскольку это связано с вашей аппаратной средой, иногда это вызывает проблемы.
* Несмотря на то, что с ним легко начать, у него крутая кривая обучения.
* Не поддерживает распределенные вычисления (поддерживается в Caffe2).
* Недостаточно хорошо документировано.

**Вывод**

Платформа глубокого обучения Caffe постоянно развивается, поскольку она имеет открытый исходный код и хорошо документирована. Его репозиторий Github был разветвлен многими разработчиками. Таким образом, в него было внесено много существенных изменений. Недавно был разработан Caffe 2, который интегрирован с репозиторием PyTorch deep learning GitHub.

### 2.5.4. Keras

Keras — это высокоуровневая библиотека с открытым исходным кодом для нейронной сети, построенная на Python, которая может быть запущена на Theano, CNTK или TensorFlow. Он был создан Франко ис Шолле, одним из инженеров Google. Он расширяем, удобен в использовании и масштабируем для более быстрого экспериментирования с нейронными сетями. Он поддерживает как CNN по отдельности, так и их комбинацию. Они не только поддерживают ЦНС. Он не способен обрабатывать низкоуровневые вычисления, поэтому для его решения используется серверная библиотека. Эта серверная библиотека используется для оболочки API, которая может работать на TensorFlow, Theano или CNTK. Серверная библиотека — это низкоуровневый API. Сначала в нем участвовало более 4800 участников, а сейчас в нем 250 000 разработчиков. Он растет так быстро, что вырос в 2 раза. Крупные компании, такие как Microsoft, NVIDIA, Google и Amazon, активно способствовали росту Keras. Индустрия хорошо интегрируется и используется для роста таких известных компаний, как Google, Netflix, Uber, Expedia и др.

**Принципы Keras**

Он был разработан для быстрой и простой работы с Python. API был "создан для людей, а не для компьютеров" и "соответствует лучшим практикам снижения когнитивного давления". Все автономные модули, которые могут быть объединены для построения новых моделей, - это нейронные слои, функции затрат, схемы оптимизации, функции активации и схемы регуляризации. По мере появления новых групп и функций можно быстро внедрять новые модули. Модели, а не отдельные файлы конфигурации модели, указываются в коде Python.

Пользовательский интерфейс Keras:

* Он не предназначен для машин: в большинстве распространенных приложений он обеспечивает прямую обратную связь при любой ошибке, которая ограничивает нет. действий пользователя.
* Высокая гибкость: включение языков глубокого обучения, таких как Theano и TensorFlow, гарантирует, что все, написанное на базовом языке, может быть реализовано в Keras, придавая ему высокую универсальность для всех его разработчиков.
* API, предназначенный для людей: Keras внедряет лучшие практики для снижения когнитивной нагрузки, поддержания точности моделей и правильных API.

**Серверная часть Keras**

Keras - отличная библиотека, предоставляющая высокоуровневые строительные блоки для разработки модели глубокого обучения. Keras не управляет всеми низкоуровневыми вычислениями, такими как продукты натяжения, свертки и т.д., А скорее полагается на специализированную библиотеку тензорных манипуляций, которая лучше оптимизирована для работы в качестве серверного движка. Keras прекрасно справился с тем, что он обеспечивает связь многих внутренних движков с Keras вместо интеграции единой библиотеки тензоров и операций, связанных с этой конкретной библиотекой.

Keras состоит из следующих трех восходящих двигателей:

* **CNTK:** Microsoft Cognitive Toolkit - это платформа с открытым исходным кодом для глубокого обучения. Он состоит из всех основных компонентов, необходимых для формирования нейронной сети. Модели основаны на C++ или Python, но для прогнозирования теста они используют C # или Java.
* **TensorFlow:** TensorFlow - это продукт Google, который является одним из наиболее известных инструментов глубокого обучения в области машинного обучения и в области сетевых исследований. Он был выпущен под лицензией Apache 2.0 9 ноября 2015 года. Он разработан таким образом, чтобы можно было легко запускать несколько процессоров и графических процессоров, а также операционные системы смартфонов. Это сделано из разных оболочек на разных языках, таких как Java, Python, C++,
* **Теано:** Теано был создан группой MILA в Монреальском университете, Канада. Theano - это библиотека python с открытым исходным кодом, обычно используемая для выполнения многомерных математических операций, включая numpy и scipy. Он использует графические процессоры для более быстрых вычислений и эффективно вычисляет градиенты путем автоматического построения символьных графиков. Он оказался очень полезным для нестабильных языков, поскольку он численно анализирует их, а затем вычисляет с использованием высокостабильных алгоритмов.

**Особенности Keras**

Некоторые из функций приведены ниже:

* Это мульти-серверная часть и поддерживает мультиплатформенность, которая позволяет кодировать все кодеры.
* Он предоставляет свободу для создания любой архитектуры, которая впоследствии используется в качестве API проекта.
* На самом деле Keras уникален тем, что легко разрабатывает модели.
* Исследовательская группа Keras очень хорошо вписывается в сообщество разработчиков.
* Это мульти-серверная часть, поддерживающая мультиплатформенность, которая поддерживает совместное кодирование всех кодировщиков.

**Преимущества:**

* Быстрое начало работы: вы можете перейти на свой собственный набор данных, не написав много кода.
* Выдающаяся производительность и быстрое прототипирование.
* Быстрая тренировка.

**Недостатки:**

* Для развертывания вам необходимо выполнить компиляцию из исходного кода, а поскольку это связано с вашей аппаратной средой, иногда это вызывает проблемы.
* Несмотря на то, что с ним легко начать, у него крутая кривая обучения.
* Не поддерживает распределенные вычисления (поддерживается в Caffe2).
* Недостаточно хорошо документировано.

**Выводы**

На основании вышеизложеннго можно сделать вывод, что наиболее рациональным подходом явяляется применением сред разработки TensorFlow и Keras для достижения цели проектирования.

## 2.6. Анализ и выбор функций активации

Основной задачей любой функции активации в любой модели, основанной на нейронной сети, является сопоставление входных данных с выходными, где входное значение получается путем вычисления взвешенной суммы входных данных нейрона и дальнейшего добавления к нему смещения (если есть смещение). Другими словами, функция активации решает, будет ли нейрон срабатывать или нет для данного входного сигнала, создавая соответствующий выходной сигнал.

В архитектуре CNN после каждого обучаемого слоя (слоя с весами, т.е. сверточного и FC-слоев) используются нелинейные слои активации. Нелинейное поведение этих слоев позволяет модели CNN изучать более сложные вещи и управлять нелинейным отображением входных данных на выходные данные. Важной особенностью функции активации является то, что она должна быть дифференцируемой, чтобы обеспечить возможность обратного распространения ошибок для обучения модели. Наиболее часто используемые функции активации в глубоких нейронных сетях (включая CNN) описаны ниже.

### 2.6.1. Sigmoid

Функция активации сигмовидной мышцы принимает действительные числа в качестве входных данных и привязывает выходные данные в диапазоне [0,1]. Кривая сигмовидной функции имеет S-образную форму. Частные случаи сигмоидной функции включают кривую Гомперца (используется в системах моделирования, которые насыщаются при больших значениях x) и кривую ogee (используется в водосбросе некоторых плотин). Сигмовидные функции имеют область всех действительных чисел, при этом возвращаемое (ответное) значение обычно монотонно увеличивается, но может уменьшаться. Сигмовидные функции чаще всего показывают возвращаемое значение (ось y) в диапазоне от 0 до 1. Другой часто используемый диапазон - от -1 до 1. Математическое представление сигмовидной мышцы таково:

(2.1)

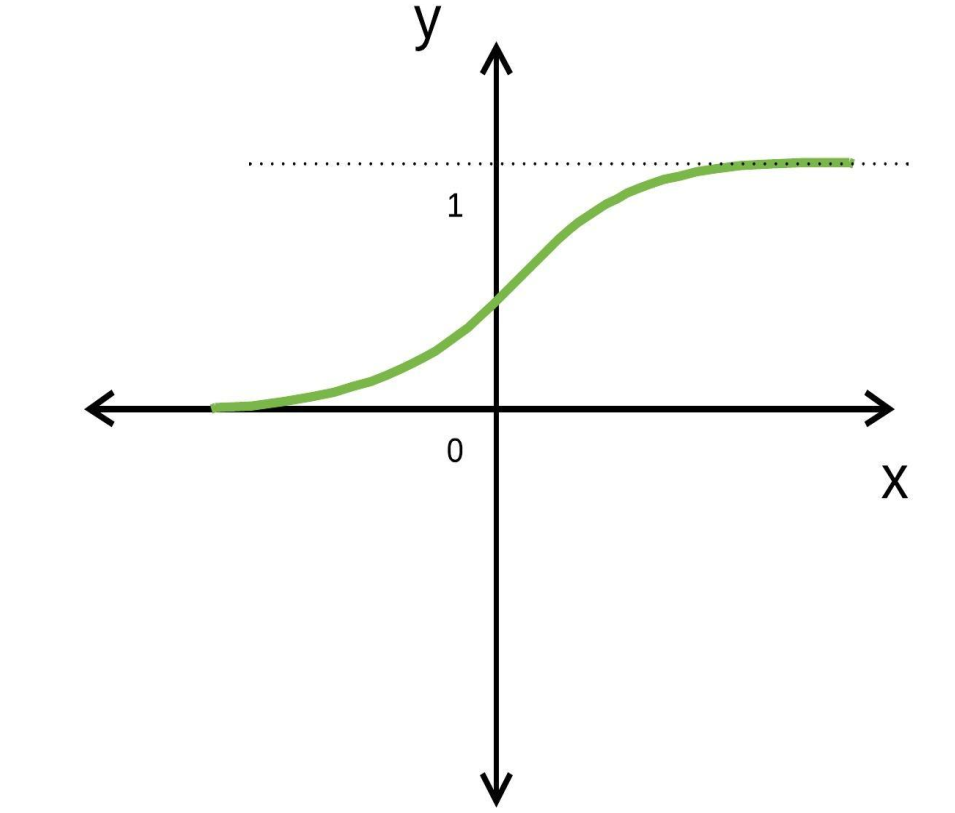


Рисунок 2.16. Кривая характеристик sigmod

### 2.6.2. Tanh

Функция активации T anh используется для привязки входных значений (действительных чисел) в диапазоне [-1, 1]. Выходной диапазон функции tanh равен (-1, 1) и имеет аналогичное поведение с сигмоидной функцией. Основным отличием является тот факт, что функция tanh переводит входные значения в 1 и -1 вместо 1 и 0. Как сигмовидные, так и функции tanh широко используются в нейронных сетях, поскольку они могут изучать очень сложные структуры. Теперь давайте сравним их, представив их сходства и различия. Математическое представление Tanh является:

(2.2)

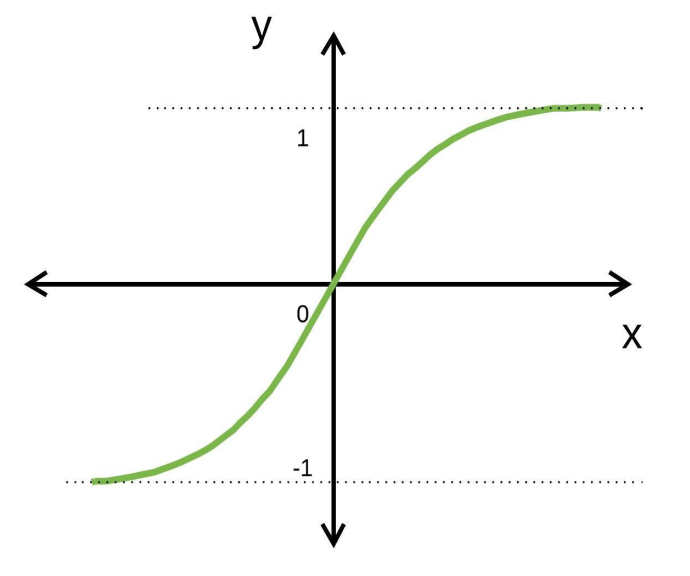


Рисунок 2.17. Кривая характеристик Tanh

### 2.6.3. ReLU

Линейный блок выпрямителя (ReLU) является наиболее часто используемой функцией активации в сверточных нейронных сетях. Он используется для преобразования всех входных значений в положительные числа. Преимущество ReLU заключается в том, что он требует очень минимальной вычислительной нагрузки по сравнению с другими. Математическое представление ReLU является:

(2.3)

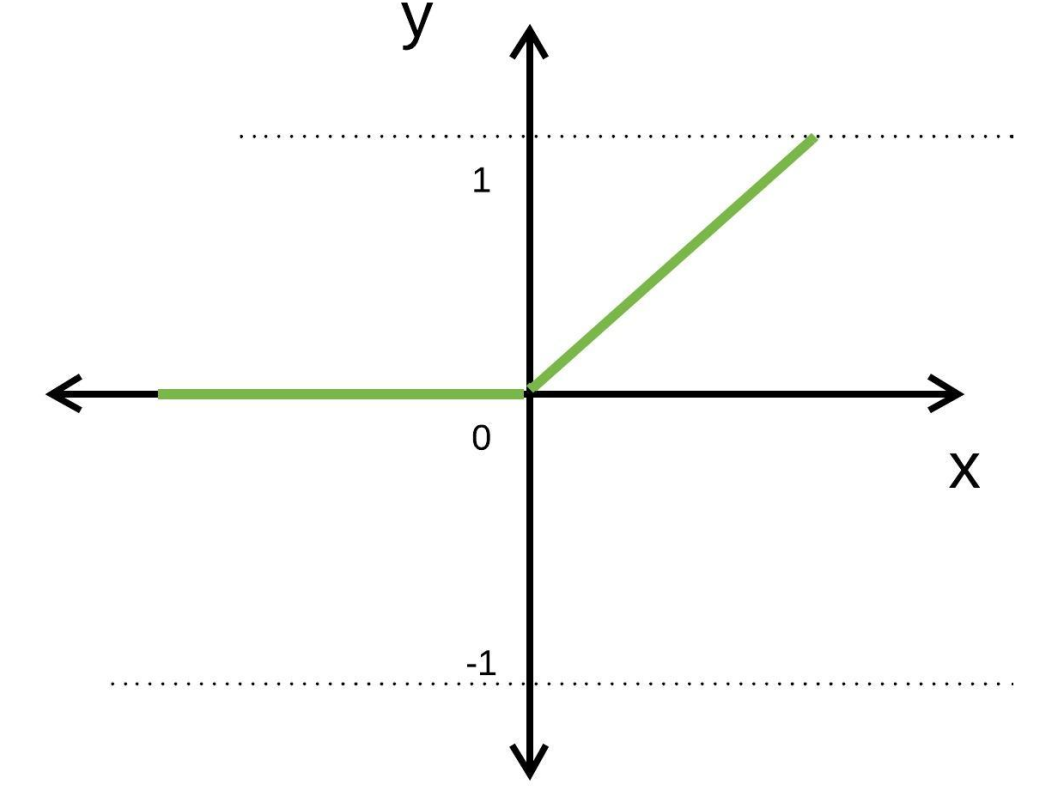


Рисунок 2.18. Кривая характеристик ReLU

Но иногда могут возникнуть некоторые серьезные проблемы при использовании функции активации ReLU. Например, рассмотрим, что больший градиент протекает во время алгоритма обратного распространения ошибок, и когда этот больший градиент передается через функцию ReLU, это может привести к обновлению весов таким образом, что нейрон никогда больше не активируется. Эта проблема известна как проблема умирающего ReLU. Для решения подобных проблем доступно несколько вариантов ReLU, некоторые из них обсуждаются ниже.

### 2.6.4. Leaky ReLU

В отличие от ReLU, функция активации Leaky ReLU с утечкой не игнорирует отрицательные входные данные полностью, а не уменьшает масштаб этих отрицательных входных данных. Негерметичный ReLU используется для решения проблемы умирающего ReLU. Математическое представление негерметичного ReLU является:

(2.4)

где m - константа, называемая коэффициентом утечки, и обычно она устанавливается на небольшое значение (например, 0,001).

Шумный ReLU используется гауссово распределение, чтобы сделать ReLU шумным. Математическое представление зашумленного ReLU является:

(2.5)

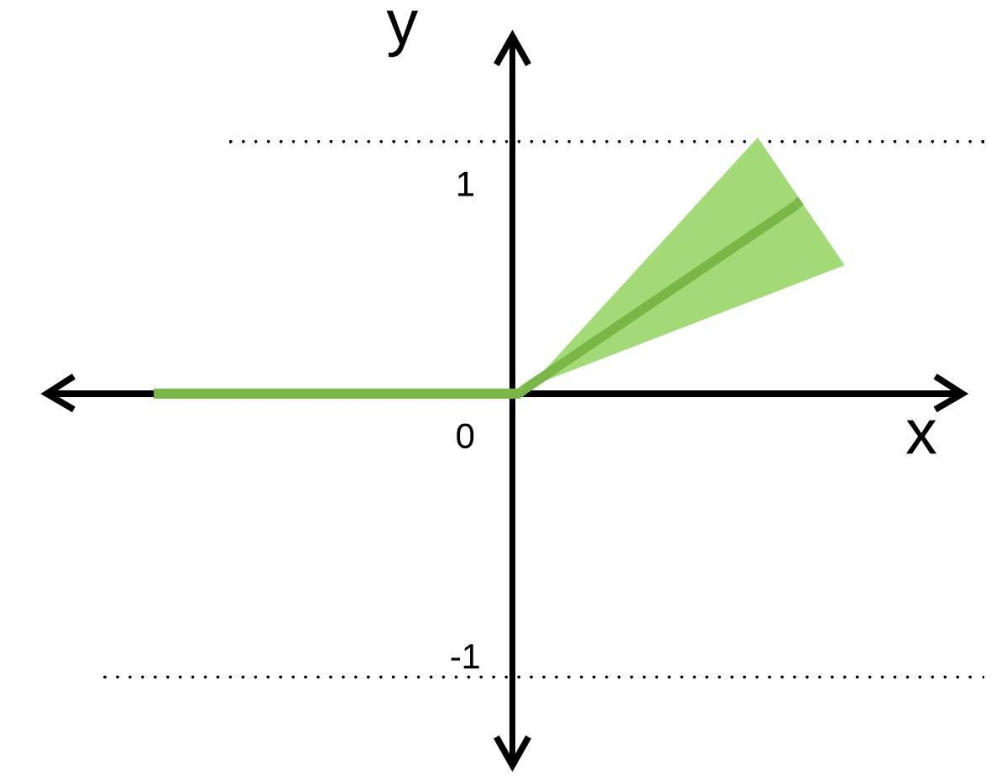


Рисунок 2.19. Кривая характеристик Noisy ReLU

### 2.6.5. Softmax

Softmax - это математическая функция, которая преобразует вектор чисел в вектор вероятностей, где вероятности каждого значения пропорциональны относительному масштабу каждого значения в векторе.

Наиболее распространенным применением функции softmax в прикладном машинном обучении является ее использование в качестве функции активации в модели нейронной сети. В частности, сеть настроена на вывод N значений, по одному для каждого класса в задаче классификации, а функция softmax используется для нормализации выходных данных, преобразуя их из значений взвешенной суммы в вероятности, сумма которых равна единице. Каждое значение в выходных данных функции softmax интерпретируется как вероятность принадлежности к каждому классу.

Функция softmax определяется следующей формулой:

(2.6)

Эта функция выводит последовательность значений вероятности, что делает ее полезной для задач многоклассовой классификации.

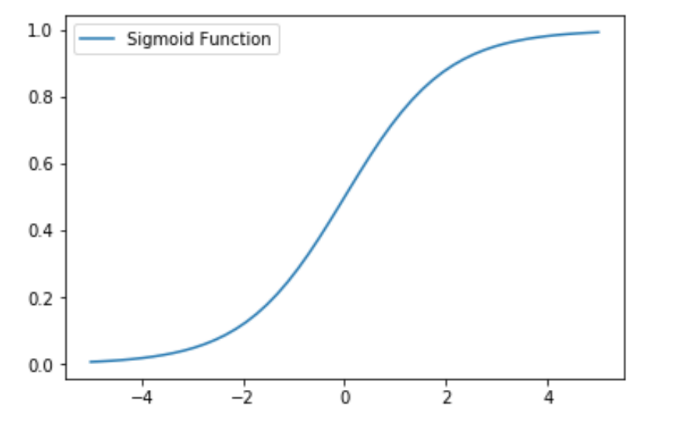


Рисунок 2.20. Кривая характеристик Softmax

**Вывод**

На основании вышеизложеннго можно сделать вывод, что наиболее рациональным подходом явяляется применение функций активации Leaky ReLU и Softmax.

## 2.7. Анализ и выбор методов инициализации весов НС

Цель инициализации веса состоит в том, чтобы предотвратить взрыв или исчезновение выходных данных активации слоя во время обучения технике DL. Обучение сети без инициализации полезного веса может привести к очень медленной конвергенции или невозможности конвергенции.

Построение даже простой нейронной сети может быть запутанной задачей, и после этого ее настройка для получения лучшего результата чрезвычайно утомительна. Но первым шагом, который учитывается при построении нейронной сети, является инициализация параметров, если все сделано правильно, то оптимизация будет достигнута за наименьшее время, в противном случае сходимость к минимумам с использованием градиентного спуска будет невозможна.

### 2.7.1. Инициализация со всеми нулями

Этот метод устанавливает все веса равными нулям (соответственно константе). Кроме того, все активации во всех нейронах одинаковы, и, следовательно, все вычисления одинаковы, что делает соответствующую модель линейной моделью.

Если все веса инициализированы 0, производная по функции потерь одинакова для каждого w в W [l], таким образом, все веса имеют одинаковое значение на последующих итерациях. Это делает скрытые единицы симметричными и продолжается для всех n итераций, т.е. Установка весов в 0 не делает ее лучше линейной модели. Важно иметь в виду, что смещения не имеют никакого эффекта, когда они инициализируются с помощью 0. Очевидно, что нулевая инициализация не является успешной при классификации.

*W[l] = np.random.zeros((l-1,l))* (2.7)

### 2.7.2. Случайная инициализация

Этот метод улучшает процесс нарушения симметрии и обеспечивает гораздо большую точность. Веса инициализируются очень близко к нулю и случайным образом. Этот метод предотвращает изучение одной и той же функции для входных параметров.

Если веса инициализируются с очень высокими значениями, термин np.dot(W, X) + b становится значительно выше, и если применяется функция активации, такая как sigmoid(), функция отображает свое значение близко к 1, где наклон градиента меняется медленно, а обучение занимает много времени.

Если веса инициализируются с низкими значениями, они сопоставляются с 0, где случай такой же, как и выше. Эту проблему часто называют исчезающим градиентом.

*W[l] = np.random.randn(l-1,l)\*10* (2.8)

Инициализация LeCun направлена на предотвращение исчезновения или взрыва градиентов во время обратного распространения путем решения растущей дисперсии с количеством входных данных и путем установки постоянной дисперсии.

Ксавье предложил более простой метод, в котором веса, такие как дисперсия активаций, одинаковы на каждом слое. Это предотвратит взрыв или исчезновение градиента.

*W[l]=np.random.randn(size\_l,size\_l-1)\*np.sqrt(1/size\_l-1)* (2.9)

### 2.7.3. Инициализация He

Эта инициализация сохраняет нелинейность функций активации, таких как повторные активации. Используя метод He, мы можем уменьшать или увеличивать значения входных данных экспоненциально.

*W[l]=np.random.randn(size\_l,size\_l-1)\*np.sqrt(2/size\_l-1)* (2.10)

Таблица 3. Преимущества и недостатки весовых методов

|  |  |  |
| --- | --- | --- |
| **Способ инициализации** | **преимущества** | **недостатки** |
| **Все-нули /**  **постоянный** | Простота | Проблема симметрии, приводящая  нейроны к изучению одних и тех же  функций |
| **Случайный** | Улучшает процесс нарушения  симметрии | - Может произойти насыщение  , приводящее к исчезновению градиента  - Уклон или уклон  невелики, что может привести  к медленному спуску по градиенту |
| **LeCun** | Решает проблемы растущей  дисперсии и  градиента | - Не используется в  сетях постоянной ширины  - Учитывает  прямое распространение  входного сигнала  - Этот метод бесполезен  , когда функция активации  недифференцируема |
| **Xavier** | Уменьшает  вероятность  проблемы исчезновения/взрыва градиента | - Этот метод бесполезен  , когда функция активации  недифференцируема  - Проблема умирающего нейрона во  время тренировки |
| **He** | Решает  проблемы умирающих нейронов | - Этот метод не полезен для  слоев с дифференцируемой  функцией активации, таких как  РеЛУ или дырявый релу |

**Выводы**

На основании вышеизложенного можно сжелать вывод что наиболее рациональным явялется использование случайного метода инициализации весов.

## 2.8. Анализ и выбор функции потерь НС

Функция потерь — это метод оценки того, насколько хорошо конкретный алгоритм моделирует данные. Если прогнозы слишком сильно отклоняются от фактических результатов, функция потерь выдаст очень большое число. Постепенно, с помощью некоторой функции оптимизации, функция потерь учится уменьшать ошибку в прогнозировании. В этой статье мы рассмотрим несколько функций потерь и их приложения в области машинного / глубокого обучения.

В алгоритмах машинного обучения нет универсальной функции потерь для всех. Существуют различные факторы, влияющие на выбор функции потерь для конкретной задачи, такие как тип выбранного алгоритма машинного обучения, простота вычисления производных и, в некоторой степени, процент выбросов в наборе данных.

В широком смысле функции потерь можно разделить на две основные категории в зависимости от типа учебной задачи, с которой мы имеем дело, — потери при регрессии и потери при классификации. При классификации мы пытаемся спрогнозировать выходные данные из набора конечных категориальных значений, т.е. Учитывая большой набор данных изображений цифр, написанных от руки, классифицируя их по одному из 0-9 цифр. Регрессия, с другой стороны, имеет дело с прогнозированием непрерывного значения, например, заданной площади этажа, количества комнат, размера комнат, прогноза цены комнаты.

### 2.8.1. Регрессионные потери

Среднеквадратичная ошибка измеряется как среднее значение квадрата разницы между прогнозами и фактическими наблюдениями. Это касается только средней величины ошибки, независимо от их направления. Однако из-за возведения в квадрат прогнозы, которые далеки от фактических значений, сильно наказываются по сравнению с менее отклоняющимися прогнозами. Кроме того, MSE обладает хорошими математическими свойствами, что облегчает вычисление градиентов.

(2.11)

**Средняя Абсолютная Погрешность**

Средняя абсолютная ошибка, с другой стороны, измеряется как среднее значение суммы абсолютных различий между прогнозами и фактическими наблюдениями. Как и MSE, это также измеряет величину ошибки без учета их направления. В отличие от MSE, MAE нуждается в более сложных инструментах, таких как линейное программирование, для вычисления градиентов. Кроме того, MAE более устойчив к выбросам, поскольку он не использует square.

(2.13)

**Средняя Ошибка Смещения**

Это гораздо менее распространено в области машинного обучения по сравнению с его аналогом. Это то же самое, что и MSE, с той лишь разницей, что мы не принимаем абсолютные значения. Очевидно, что необходимо соблюдать осторожность, поскольку положительные и отрицательные ошибки могут компенсировать друг друга. Хотя на практике он менее точен, он может определить, имеет ли модель положительное смещение или отрицательное смещение.

(2.14)

### 2.8.2. Классификационные Потери

Проще говоря, оценка правильной категории должна быть больше суммы баллов всех неправильных категорий на некоторый запас прочности (обычно один). И, следовательно, потеря шарнира используется для классификации с максимальным запасом, особенно для машин с опорными векторами. Хотя это и не дифференцируемая, это выпуклая функция, которая упрощает работу с обычными выпуклыми оптимизаторами, используемыми в области машинного обучения.

(2.15)

**Перекрестная Потеря Энтропии**

Это наиболее распространенная настройка для задач классификации. Кросс-энтропийные потери увеличиваются по мере того, как прогнозируемая вероятность расходится с фактической меткой. Кросс-энтропия вычислит оценку, которая суммирует среднюю разницу между фактическим и прогнозируемым распределением вероятностей для прогнозирования класса 1. Оценка сведена к минимуму, а идеальное значение кросс-энтропии равно 0.

(2.16)

Обратите внимание, что, когда фактическая метка равна 1 (y(i) = 1), вторая половина функции исчезает, тогда как в случае, если фактическая метка равна 0 (y(i) = 0), первая половина отбрасывается. Короче говоря, мы просто умножаем логарифм фактической прогнозируемой вероятности для основного класса истинности. Важным аспектом этого является то, что потеря перекрестной энтропии сильно наказывает прогнозы, которые являются уверенными, но ошибочными.

## 2.9. Анализ и выбор показателей качества (метрик) НС

Метрика отличается от функции потерь. Функции потерь — это функции, которые показывают меру производительности модели и используются для обучения модели машинного обучения (с использованием какой-либо оптимизации) и обычно дифференцируются по параметрам модели. С другой стороны, метрики используются для мониторинга и измерения производительности модели (во время обучения и тестирования) и не обязательно должны быть дифференцируемыми. Однако, если для некоторых задач метрика производительности дифференцируема, ее можно использовать как функцию потерь (возможно, с добавлением к ней некоторых регуляризаций), так и метрику, такую как MSE.

**Точность классификации**

Точность классификации, пожалуй, является самой простой метрикой, которую можно себе представить, и определяется как количество правильных прогнозов, деленное на общее количество прогнозов, умноженное на 100. Таким образом, в приведенном выше примере из 1100 выборок 1030 предсказаны правильно, что приводит к точности классификации в:

*Точность классификации = True\_Positive+True\_Negative / (True\_Positive+ False\_Positive+True\_Negative+False\_Negative) = (90+940)/ (1000+100) = 1030/1100= 93.6%* (2.17)

**Точность**

Есть много случаев, когда точность классификации не является хорошим показателем эффективности нашей модели. Один из таких сценариев - это когда наше распределение классов несбалансировано (один класс встречается чаще, чем другие). В этом случае, даже если мы прогнозируем все выборки как наиболее частый класс, мы получим высокую точность, что вообще не имеет смысла (потому что ваша модель ничему не учится, а просто предсказывает все как высший класс).

*Точность= True\_Positive/ (True\_Positive+ False\_Positive)* (2.18)

**Отзыв**

Отзыв - еще одна важная метрика, которая определяется как доля выборок из класса, которые правильно предсказаны моделью. Более формально:

*Отзыв= True\_Positive/ (True\_Positive+ False\_Negative)* (2.19)

**Результат Формулы-1**

В зависимости от приложения вы можете захотеть придать более высокий приоритет отзыву или точности. Но есть много приложений, в которых важны как отзыв, так и точность. Поэтому естественно подумать о том, как объединить эти два показателя в единую метрику. Одна популярная метрика, которая сочетает в себе точность и отзыв, называется F1-score, которая представляет собой среднее гармоническое значение точности и отзыва, определяемое как:

*F1-оценка= 2\*Точность\*Отзыв/(Точность+Отзыв)* (2.20)

**Чувствительность и специфичность**

Чувствительность и специфичность — это два других популярных показателя, которые в основном используются в областях, связанных с медициной и биологией, и определяются как:

*Чувствительность= Отзыв= True\_Positive / (True\_Positive+ False\_Negative)*

*Специфичность = Истинная Отрицательная скорость = True\_Negative / (True\_Negative+False\_Positive)* (2.21)

## 2.10. Анализ и выбор методов оптимизации НС

Оптимизаторы — это алгоритмы или методы, используемые для изменения атрибутов нашей нейронной сети, таких как веса и скорость обучения, чтобы уменьшить потери.

**Gradient Descent**

Gradient Descent - самый простой, но наиболее часто используемый алгоритм оптимизации. Он широко используется в алгоритмах линейной регрессии и классификации. Обратное распространение в нейронных сетях также использует алгоритм градиентного спуска.

Градиентный спуск — это алгоритм оптимизации первого порядка, который зависит от производной первого порядка функции потерь. Он вычисляет, в какую сторону следует изменить веса, чтобы функция могла достичь минимумов. Посредством обратного распространения потери передаются с одного уровня на другой, и параметры модели, также известные как веса, изменяются в зависимости от потерь, чтобы потери можно было свести к минимуму.

*алгоритм: θ=θ−α⋅∇J(θ)* (2.22)

**Преимущества**

* Простое вычисление.
* Легко внедряется.
* Легко понять.

**Недостатки**

* Может оказаться в ловушке на локальных минимумах.
* Веса изменяются после вычисления градиента для всего набора данных. Таким образом, если набор данных слишком велик, то для достижения минимумов могут потребоваться годы.
* Требуется большая память для вычисления градиента по всему набору данных.

**Stochastic Gradient Descent**

Это вариант градиентного спуска. Он пытается чаще обновлять параметры модели. При этом параметры модели изменяются после вычисления потерь в каждом обучающем примере. Таким образом, если набор данных содержит 1000 строк, SGD обновит параметры модели 1000 раз за один цикл набора данных вместо одного раза, как при градиентном спуске.

*θ=θ−α⋅∇J (θ; x(i); y(i))* (2.23)

где {x(i), y(i)} - обучающие примеры.

Поскольку параметры модели часто обновляются, параметры имеют высокую дисперсию и колебания функций потерь при различной интенсивности.

**Преимущества**

* Следовательно, частое обновление параметров модели приводит к сходимости за меньшее время.
* Требуется меньше памяти, так как нет необходимости хранить значения функций потерь.
* Могут появиться новые минимумы.

**Недостатки**

* Высокая дисперсия параметров модели.
* Может выстрелить даже после достижения глобальных минимумов.
* Чтобы получить ту же сходимость, что и при градиентном спуске, необходимо медленно снижать значение скорости обучения.

**Mini-Batch Gradient Descent**

Это лучший из всех вариантов алгоритмов градиентного спуска. Это улучшение как по сравнению с SGD, так и по сравнению со стандартным градиентным спуском. Он обновляет параметры модели после каждой партии. Таким образом, набор данных делится на различные пакеты, и после каждого пакета параметры обновляются.

*θ=θ−α⋅∇J(θ; B(i))* (2.24)

где {B(i)} - партии обучающих примеров.

**Преимущества**

* Часто обновляет параметры модели, а также имеет меньшую дисперсию.
* Требуется средний объем памяти.

**Недостатки**

* Все типы градиентного спуска имеют некоторые проблемы
* Выбор оптимального значения скорости обучения. Если скорость обучения слишком мала, то для сходимости градиентного спуска может потребоваться целая вечность.
* Иметь постоянную скорость обучения для всех параметров. Могут быть некоторые параметры, которые мы, возможно, не захотим изменять с той же скоростью.
* Может попасть в ловушку на локальных минимумах.

**Momentum**

Momentum был изобретен для уменьшения высокой дисперсии в SGD и смягчения сходимости. Это ускоряет конвергенцию в соответствующем направлении и уменьшает флуктуацию в нерелевантном направлении. В этом методе используется еще один гиперпараметр, известный как импульс, обозначаемый символом ‘γ’.

*V(t)=yV(t−1) +α.∇J(θ)* (2.25)

Теперь веса обновляются на θ=θ−V(t).

Член импульса γ обычно устанавливается равным 0,9 или аналогичному значению.

**Преимущества**

* Уменьшает колебания и высокую дисперсию параметров.
* Сходится быстрее, чем градиентный спуск.

**Недостатки**

* Добавлен еще один гиперпараметр, который необходимо выбрать вручную и точно.

**Nesterov Accelerated Gradient**

Импульс может быть хорошим методом, но если импульс слишком высок, алгоритм может пропустить локальные минимумы и может продолжить расти. Итак, для решения этой проблемы был разработан алгоритм NAG. Это метод заглядывания вперед. Мы знаем, что будем использовать yV(t−1) для изменения весов, поэтому θ−yV(t−1) приблизительно сообщает нам будущее местоположение. Теперь мы рассчитаем стоимость на основе этого будущего параметра, а не текущего.

*V(t)=yV(t−1)+α. ∇J( θ−yV(t−1))* (2.26)

а затем обновите параметры, используя θ=θ−V(t).

**Преимущества**

* Не пропускает локальные минимумы.
* Замедляется, если возникают минимумы.

**Недостатки**

* Тем не менее, гиперпараметр необходимо выбрать вручную.

**Adagrad**

Одним из недостатков всех объясненных оптимизаторов является то, что скорость обучения постоянна для всех параметров и для каждого цикла. Этот оптимизатор изменяет скорость обучения. Он изменяет скорость обучения ‘η’ для каждого параметра и на каждом временном шаге ‘t’. Это тип алгоритма оптимизации второго порядка. Он работает на производной функции ошибки.

(2.27)

Производная функции потерь для заданных параметров в данный момент времени t.

(2.28)

Обновляйте параметры для заданных входных данных i и вовремя/итерации t

η — это скорость обучения, которая изменяется для данного параметра θ(i) в данный момент времени на основе предыдущих градиентов, вычисленных для данного параметра θ(i).

Мы сохраняем сумму квадратов градиентов w.r.t. θ(i) до временного шага t, в то время как ϵ — это сглаживающий член, который позволяет избежать деления на ноль (обычно порядка 1e-8). Интересно, что без операции с квадратным корнем алгоритм работает намного хуже. Он делает большие обновления для менее частых параметров и небольшой шаг для частых параметров.

**Преимущества**

* Скорость обучения изменяется для каждого параметра обучения.
* Не нужно вручную настраивать скорость обучения.
* Способен тренироваться на разреженных данных.

**Недостатки**

* Вычислительно дорого из-за необходимости вычисления производной второго порядка.
* Скорость обучения всегда снижается, что приводит к медленному обучению.

**AdaDelta**

Это расширение AdaGrad, которое, как правило, устраняет проблему замедления скорости обучения. Вместо того, чтобы накапливать все ранее возведенные в квадрат градиенты, Adadelta ограничивает окно накопленных прошлых градиентов некоторым фиксированным размером w. При этом используется экспоненциально скользящая средняя, а не сумма всех градиентов.

*E[g2] (t)=γ.E[g2](t−1)+(1−γ).g2(t)* (2.29)

Мы устанавливаем γ на то же значение, что и член импульса, около 0,9.

**Преимущества**

* Теперь скорость обучения не снижается, и обучение не прекращается.

**Недостатки:**

* Вычислительно дорого.

**Adam**

Adam (Адаптивная оценка момента) работает с импульсами первого и второго порядка. Интуиция, стоящая за Adam, заключается в том, что мы не хотим катиться так быстро только потому, что можем перепрыгнуть через минимум, мы хотим немного уменьшить скорость для тщательного поиска. В дополнение к хранению экспоненциально убывающего среднего значения прошлых квадратов градиентов, таких как AdaDelta, Adam также сохраняет экспоненциально убывающее среднее значение прошлых градиентов M (t).

M (t) и V (t) являются значениями первого момента, который является средним, и второго момента, который является нецентрированной дисперсией градиентов соответственно.

(2.30)

(2.31)

Первый и второй порядок импульса

Здесь мы берем среднее значение M(t) и V (t), так что E [m(t)] может быть равно E[g(t)], где E [f (x)] - ожидаемое значение f(x).

Чтобы обновить параметр:

(2.32)

Значения для β1 равны 0,9, 0,999 для β2 и (10 x exp (-8)) для ‘ϵ’.

**Преимущества:**

* Этот метод слишком быстр и быстро сходится.
* Исправляет исчезающую скорость обучения, высокую дисперсию.

**Недостатки:**

* Вычислительно затратно.

**Выводы**

На основании вышеизложеннго можно сделать вывод, что наиболее рациональным подходом явяляется применение оптимизатора Adam

## 2.11. Анализ и выбор методов регуляризации для предотвращения обучения

Регуляризация — это метод, который вносит небольшие изменения в алгоритм обучения таким образом, чтобы модель лучше обобщалась. Это, в свою очередь, также улучшает производительность модели для невидимых данных. Существует несколько различных методов для применения регуляризации в глубоком обучении.

**Регуляризация L2 и L1**

L1 и L2 являются наиболее распространенными типами регуляризации. Они обновляют общую функцию затрат, добавляя еще один термин, известный как термин регуляризации.

*Функция затрат = Потери (скажем, двоичная перекрестная энтропия) + Член регуляризации* (2.33)

Из-за добавления этого члена регуляризации значения весовых матриц уменьшаются, поскольку предполагается, что нейронная сеть с меньшими весовыми матрицами приводит к более простым моделям. Таким образом, это также в значительной степени уменьшит переобучение.

Однако этот член регуляризации отличается в L1 и L2.

В L2 у нас есть:

(2.34)

Здесь лямбда — это параметр регуляризации. Это гиперпараметр, значение которого оптимизировано для достижения лучших результатов. Регуляризация L2 также известна как уменьшение веса, поскольку она заставляет веса уменьшаться до нуля (но не точно до нуля).

В L1 мы имеем:

(2.35)

**Увеличение объема Данных**

Самый простой способ уменьшить переобучение — это увеличить размер обучающих данных. В машинном обучении мы не смогли увеличить размер обучающих данных, поскольку помеченные данные были слишком дорогостоящими. В этом случае существует несколько способов увеличения размера обучающих данных – поворот изображения, переворачивание, масштабирование, сдвиг и т.д. На приведенном ниже изображении в наборе данных рукописных цифр было произведено некоторое преобразование.



Рисунок 2.21. Преобразование набора данных рукописных цифр

Этот метод известен как увеличение данных. Обычно это обеспечивает большой скачок в повышении точности модели. Это можно рассматривать как обязательный трюк для того, чтобы улучшить наши прогнозы.

**Ранняя остановка**

Ранняя остановка — это своего рода стратегия перекрестной проверки, при которой мы сохраняем одну часть обучающего набора в качестве набора проверки. Когда мы видим, что производительность набора проверки ухудшается, мы немедленно прекращаем обучение модели. Это известно как ранняя остановка.

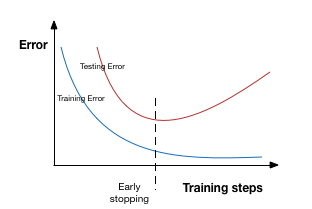


Рисунок 2.22. Способ ранней остановки

На приведенном выше изображении мы остановим обучение на пунктирной линии, так как после этого наша модель начнет переобучаться на обучающих данных.

**Dropout**

Это один из самых интересных типов методов регуляризации. Он также дает очень хорошие результаты и, следовательно, является наиболее часто используемым методом регуляризации в области глубокого обучения. На Рисунке 2.23 показана структура нейронной сети.

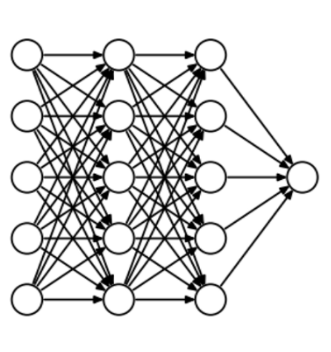


Рисунок 2.23. Структура нейронной сети

На каждой итерации dropout случайным образом выбирает некоторые узлы и удаляет их вместе со всеми их входящими и исходящими соединениями, как показано ниже.

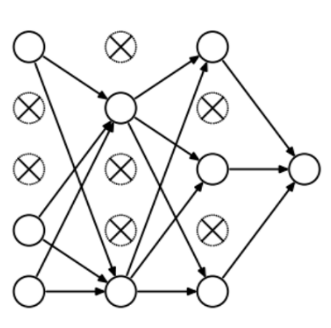


Рисунок 2.24. Структура нейронной сети после отсева

Таким образом, каждая итерация имеет другой набор узлов, и это приводит к другому набору выходных данных. Его также можно рассматривать как ансамблевую технику в машинном обучении.

Ансамблевые модели обычно работают лучше, чем одиночная модель, поскольку они улавливают больше случайности. Аналогичным образом, отсев также работает лучше, чем обычная модель нейронной сети.

Эта вероятность выбора того, сколько узлов должно быть удалено, является гиперпараметром функции отсева. Как видно на изображении выше, выпадение может быть применено как к скрытым слоям, так и к входным слоям.

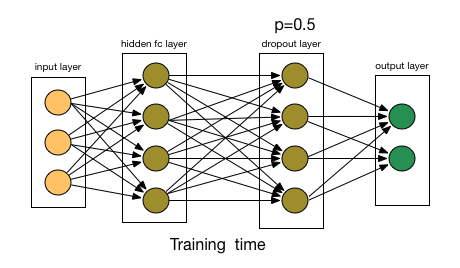


Рисунок 2.25. Отсевающий слой в нейронных сетях

**Выводы**

На основании вышеизложеннго можно сделать вывод, что наиболее рациональным подходом явяляется выбор метода регуляризации Dropout

## 2.12. Конструирование топологии НС с применением добавления/удаления дополнительных скрытых слоев и прореживания

Выбор топологии нейронной сети относится к систематической процедуре выбора между конкурирующими моделями. Естественно, это рассматривается как ключевой аспект оптимизации и надежности работы нейронной сети. При построении топологий нейронных сетей необходимо с самого начала определить общую таксономию архитектур нейронных сетей, которые будут построены. Таксономия, рассматриваемая в этом исследовании, представляет собой общую таксономию изменяющихся во времени паттернов, которая объединяет многие существующие архитектуры в литературе и указывает на несколько перспективных архитектур нейронных сетей, которые еще предстоит изучить. Контекст проблемы заключается в том, что выбор правильной топологии нейронной сети для использования в конкретной области, такой как прогнозирование корпоративного банкротства с оптимальной производительностью обобщения, в любом случае не является тривиальной проблемой. На Рисунке 2.26 показана архитектура VGG21, но я добавил несколько необходимых слоев, используя последовательный режим.

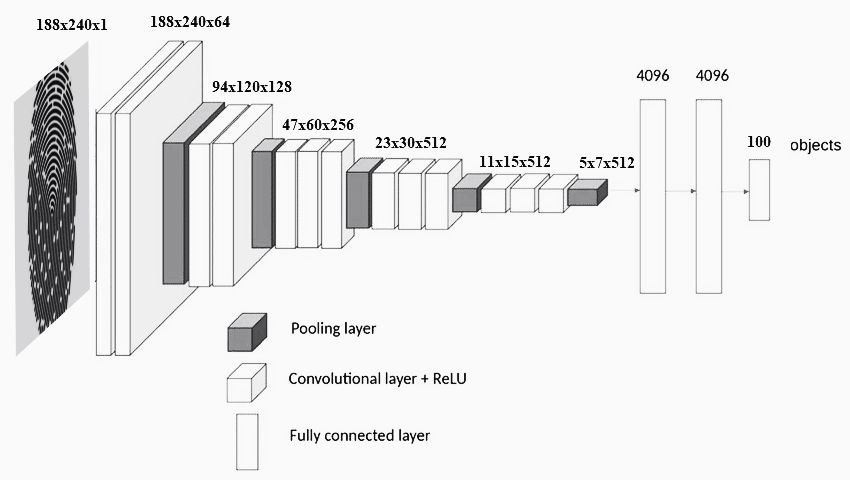


Рисунок 2.26. Архитектура VGG21

Обучение глубоких нейронных сетей - сложная задача, требующая решения нескольких проблем. Несмотря на их огромный потенциал, они могут быть медлительными и склонными к переобучению. Пакетная нормализация– обычно сокращаемая как Пакетная норма, является одним из таких методов. В настоящее время это широко используемая методика в области глубокого обучения. Это повышает скорость обучения нейронных сетей и обеспечивает регуляризацию, избегая переобучения. Пакетная норма работает очень похожим образом в сверточных нейронных сетях. Хотя мы могли бы сделать это так же, как и раньше, мы должны следовать свойству свертки. В свертках у нас есть общие фильтры, которые проходят по картам объектов входных данных (на изображениях карта объектов обычно имеет высоту и ширину). Эти фильтры одинаковы на каждой карте объектов. Затем разумно нормализовать выходные данные таким же образом, распределив их по картам объектов.

**Batch Normalization** — это метод нормализации, выполняемый между слоями нейронной сети, а не в исходных данных. Это делается по мини-пакетам вместо полного набора данных. Это служит для ускорения обучения и использования более высоких темпов обучения, что облегчает обучение. Мы можем определить формулу нормализации пакетной нормы как:

(2.36)

где mz - среднее значение выходного сигнала нейронов, а sz - стандартное отклонение выходного сигнала нейронов.

На следующем изображении мы можем видеть обычную нейронную сеть с прямой связью: xi - входные данные, z - выходные данные нейронов, a - выходные данные функций активации, а y - выходные данные сети:

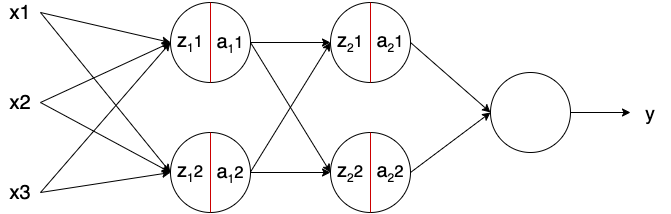


Рисунок 2.27. Нейронная сеть с прямой связью

Пакетная норма – на изображении, представленном красной линией, – применяется к выходу нейронов непосредственно перед применением функции активации. Обычно нейрон без пакетной нормы вычисляется следующим образом:

(2.36)

где g() - линейное преобразование нейрона, - веса нейрона, b - смещение нейронов и f() - функция активации. Модель запоминает параметры и b. Добавляя пакетную норму, она выглядит следующим образом:

(2.37)

где zN - выходные данные пакетной нормы, mz - среднее значение выходных данных нейронов, sz - стандартное отклонение выходных данных нейронов, а также и -параметры обучения пакетной нормы. Обратите внимание, что смещение нейронов (b) устранено. Это связано с тем, что, когда мы вычитаем среднее значение mz, любая константа над значениями z – например, b – может быть проигнорирована, поскольку она будет вычитаться сама по себе.

Параметры и смещают среднее значение и стандартное отклонение соответственно. Таким образом, выходные данные пакетной нормы по слою приводят к распределению со средним значением и стандартным отклонением . Эти значения изучаются в течение эпох, а другие параметры обучения, такие как веса нейронов, направлены на уменьшение потерь модели.

## 2.13. Структура и состав компонентов системы

Структура и состав системы показаны на pисунке 2.28. В моей системе входной отпечаток пальца должен быть изображением, и мы можем получить его со сканера, камеры или другими способами. Затем мы изменяем размер изображения. В этой системе размер изображения составляет 96x96 пикселей. Мы используем 1000 отпечатков пальцев 100 человек. Затем изображения преобразуются из изображений в оттенках серого в двоичные изображения. Бинарные изображения обучаются с использованием модели CNN. В этой системе используются предварительно обученные модели CNN. Предварительно обученная модель — это модель, созданная кем-то другим для решения аналогичной проблемы. Вместо того, чтобы создавать модель с нуля для решения аналогичной проблемы, мы можем использовать модель, обученную другой проблеме, в качестве отправной точки. Это экономит огромные усилия, необходимые для изобретения колеса. После обучения модели мы тестируем случайное изображение путем сравнения с обучающими данными. Затем мы получаем результат идентификации системы отпечатков пальцев.

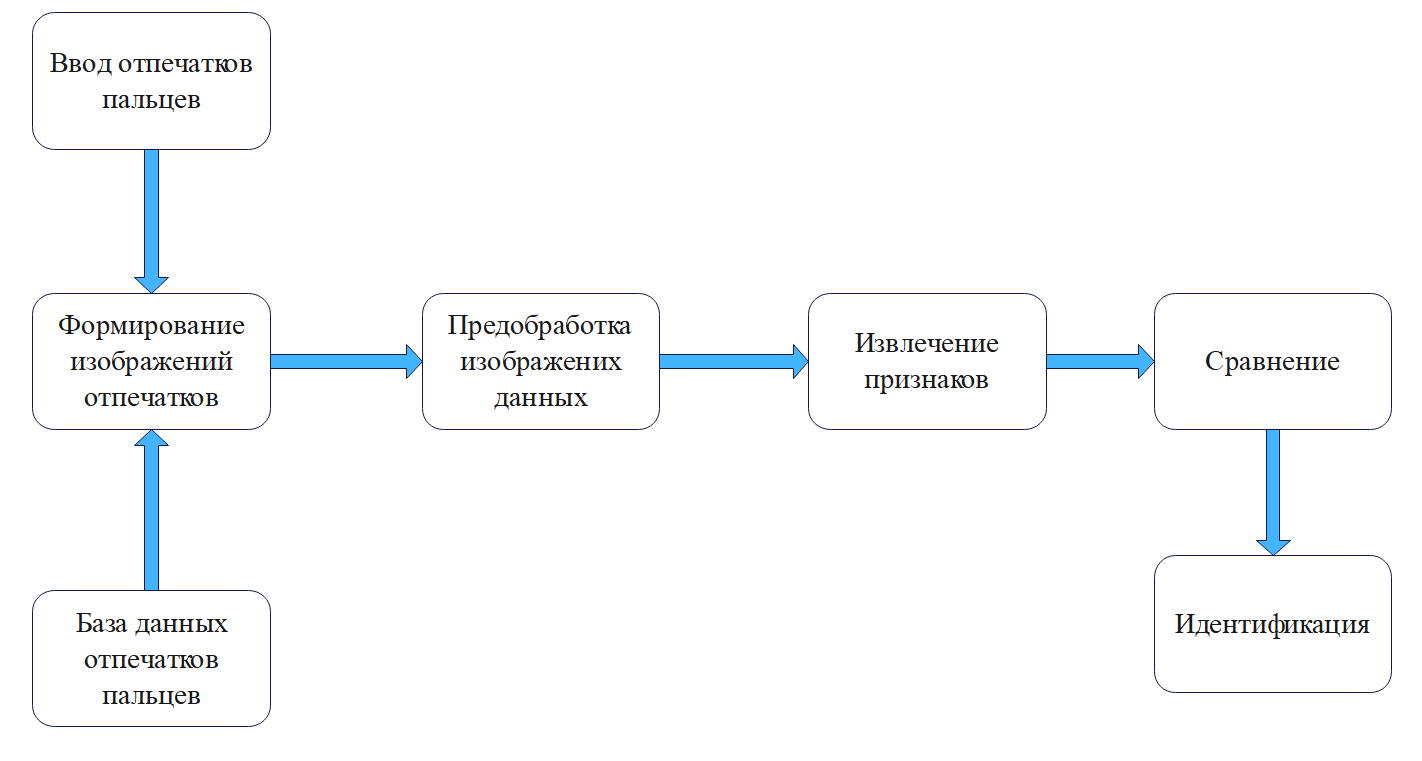


Рисунок 2.28. Топология интеллектуальной системы идентификации человека по отпечаткам пальцев

## 2.14. Структура и состав программного обеспечения системы

На Рисунке показаны структура и состав системного программного обеспечения. Система идентификации отпечатков пальцев реализована на компьютере i7 с частотой 2,8 ГГц с Windows 10 и Google Colab с использованием языка python. Набор данных отпечатков пальцев Sokoto Coventry (SO Coding) реализован как биометрическая база данных отпечатков пальцев, предназначенная для академических исследовательских целей. SOCOFing состоит из 6000 изображений отпечатков пальцев от 600 африканских субъектов (в этой системе использовалось 1000 отпечатков пальцев на 100 человек) и содержит уникальные атрибуты, такие как метки для обозначения пола, имени руки и пальца, а также синтетически измененные версии с тремя различными уровнями изменения для стирания, центрального поворота и z-cut. В системе идентификации отпечатков пальцев используется метод глубокого обучения (сверточная нейронная сеть). Система идентификации отпечатков пальцев используется во многих приложениях. Это используется в системах контроля безопасности (мобильных, компьютерных и промышленных). Они очень полезны для расследования.

## 2.15. Интерфейс взаимодействия пользователя с системой

Поскольку отпечатки пальцев имеют уникальные характерные особенности, они использовались для идентификации личности и аутентификации. Многие системы безопасности и системы управления входом основаны на технологии распознавания отпечатков пальцев. Пользовательский интерфейс отпечатков пальцев — это первая попытка применить технологию распознавания отпечатков пальцев к области пользовательского интерфейса. Она отличается от систем идентификации личности тем, что использует не только уникальность отпечатков пальцев разных людей, но и уникальность десяти пальцев одного человека. С точки зрения пользовательских интерфейсов пользовательский интерфейс отпечатков пальцев можно классифицировать как мультимодальный пользовательский интерфейс, который позволяет пользователю взаимодействовать с компьютером, используя такие модальности, как речь, жесты, выражения лица и направление взгляда. В мультимодальных системах речь была наиболее широко используемой модальностью. Это связано с тем, что это наиболее естественное средство для человеческого общения, и оно способно передавать различные типы информации. Однако интерфейс на основе речи не подходит для использования в общественных местах, поскольку он может беспокоить других и может привести к утечке содержимого работы пользователя. Хотя FUI обладает меньшими возможностями для передачи информации, чем интерфейс на основе речи, его можно использовать публично без каких-либо проблем.

На Рисунке 2.29 показана базовая архитектура пользовательского интерфейса отпечатков пальцев. Изображение отпечатка пальца захватывается любым способом, когда палец касается сканера отпечатков пальцев, встроенного в поверхность устройства ввода. Если захваченное изображение совпадает с одним из шаблонов отпечатков пальцев, зарегистрированных в базе данных finger ID, соответствующая команда или объект данных отправляются в целевую операционную систему. Обученная модель сопоставляется с входными данными изображения и выводит возможный результат отпечатка пальца.

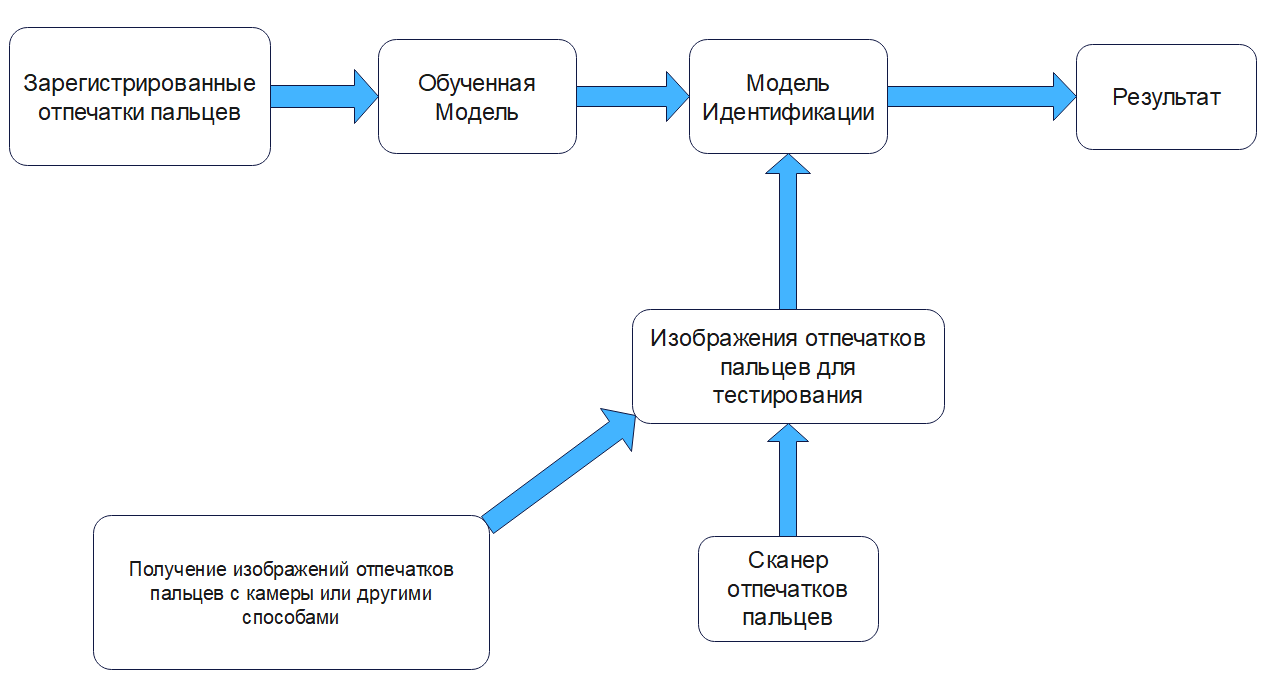


Рисунок 2.29. Интерфейс взаимодействия пользователя с системой

# 3. ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

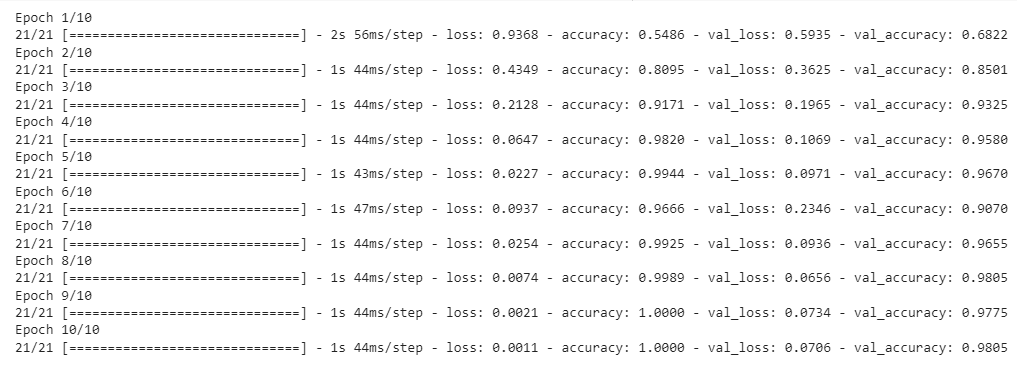
## 3.1. Экспериментальная отработка и настройка проектных решений системы (с применением модификации числа периодов обучения, числа нейронов в скрытых слоях, размера пакета)

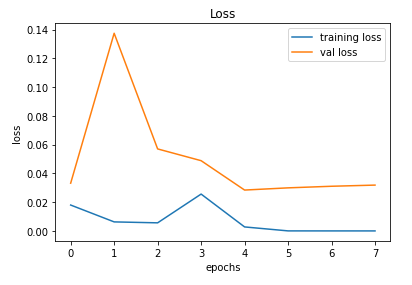
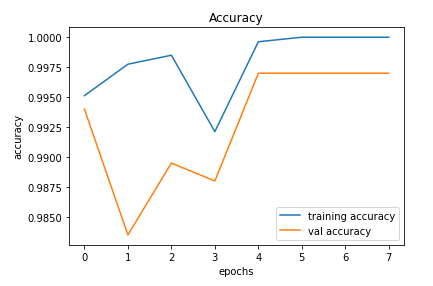
В своей диссертации я попробовал идентификацию отпечатков пальцев с использованием трех моделей глубоких сверточных нейронных сетей. Это Alexnet, VGG 16 и VGG 21. Эти модели были разработаны для достижения самых современных результатов в идентификации отпечатков пальцев. По сравнению с моделями AlexNet, VGG 16 и VGG 21 используют меньшую базу данных. В процессе обучения всех моделей использовались оптимизатор «Adam», функция потерь «categorical\_crossentropy», функция активации «softmax» и метрика «accuracy».

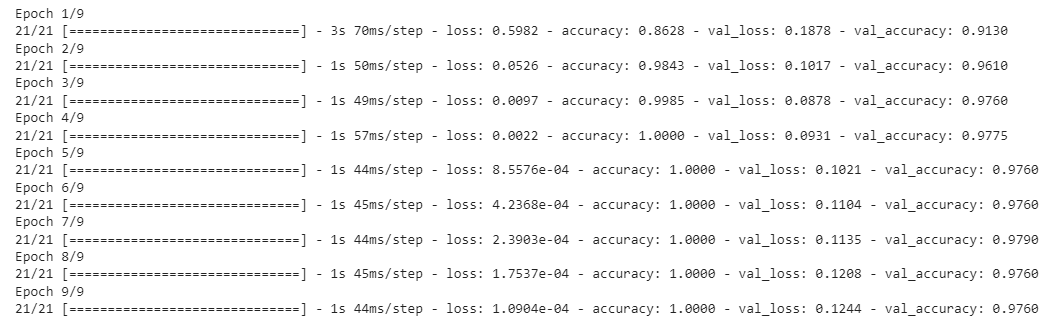
**Результаты в модели Alexnet**

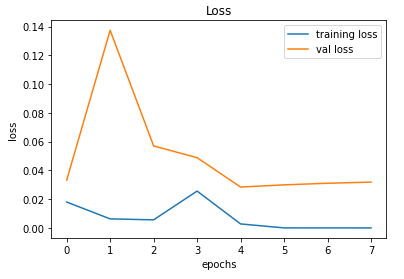
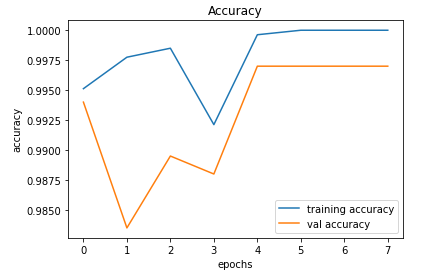
На Рисунке показана архитектура AlexNet. Архитектура AlexNet состоит из 5 сверточных слоев, 3 слоев максимального объединения, 2 слоев нормализации, 2 полностью связанных слоев и 1 слоя softmax. Каждый сверточный слой состоит из сверточных фильтров и нелинейной функции активации ReLU. Слои объединения используются для выполнения максимального объединения. Размер входного сигнала фиксирован из-за наличия полностью соединенных слоев. Размер ввода рекомендуется как 224x224x3, но в моей диссертации используется 96x96x1.

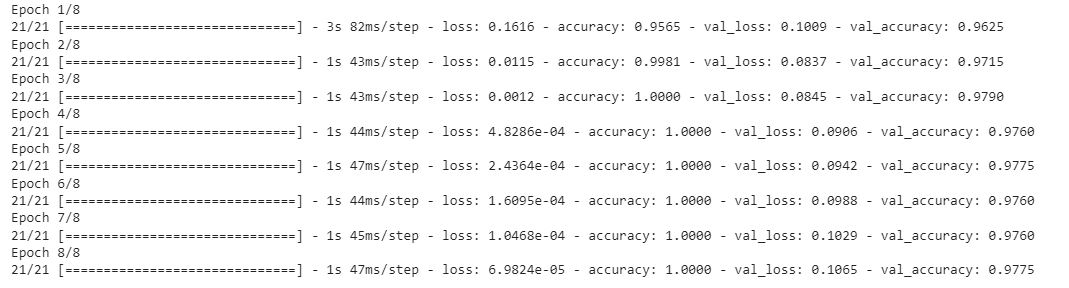
На Рисунке 2.30 показаны результаты модели AlexNet для эпох 8, 9 и 10. Нам нужно выбрать лучший результат, и поэтому требуется более высокая точность и меньшие потери. Время также необходимо как можно быстрее.











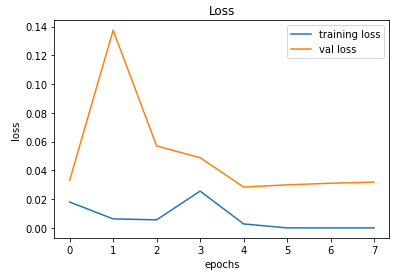
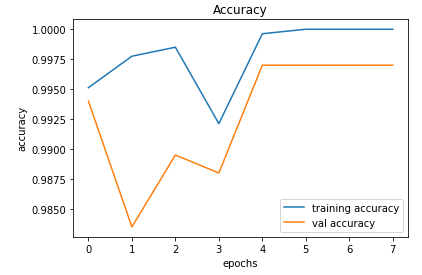


Рисунок 2.30. Показаны результаты модели AlexNet для эпох 8, 9 и 10

Точность, точность проверки показаны в таблице (4), а потери, потери при проверке показаны в таблице (5). Теперь мы выберем лучший результат.

Таблица 4. Время, точность и точность проверки AlexNet для эпох 8, 9, 10

|  |  |  |  |
| --- | --- | --- | --- |
| Эпохи | Время | Точности | Точности проверки |
| 10 | 11 | 1,0000 | 0,9805 |
| 9 | 11 | 1,0000 | 0,9760 |
| 8 | 10 | 1,0000 | 0,9775 |

Таблица 5. Потери и потери проверки системы для эпох 8, 9, 10

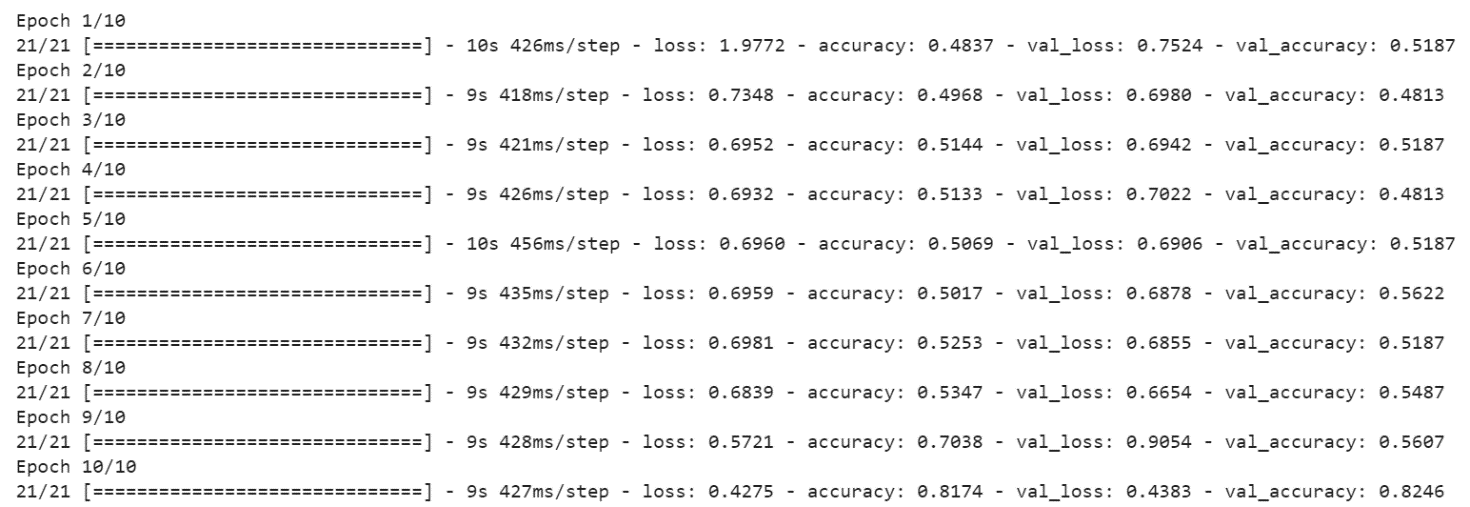
|  |  |  |
| --- | --- | --- |
| Эпохи | Потери | Потери проверки |
| 10 | 0,0011 | 0,0706 |
| 9 | 1,0904 | 0,1244 |
| 8 | 6,9824 | 0,1065 |

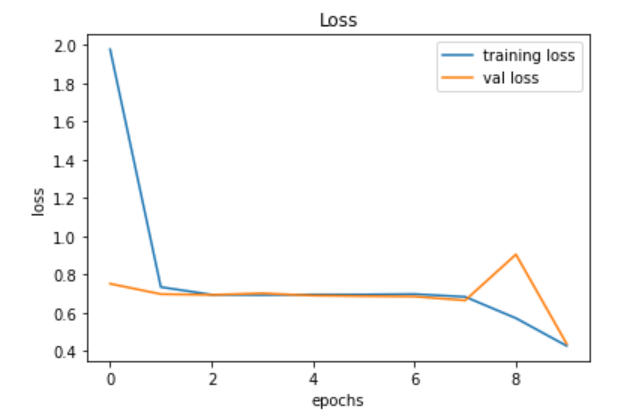
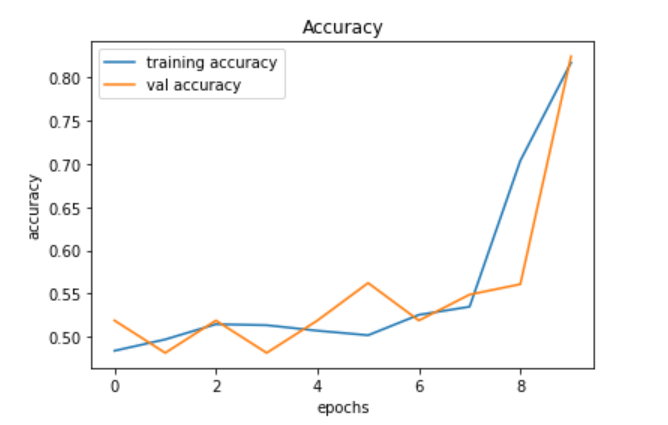
Согласно таблице (4) и таблице (5), epoch-10 обладает лучшей точностью и меньшими потерями.

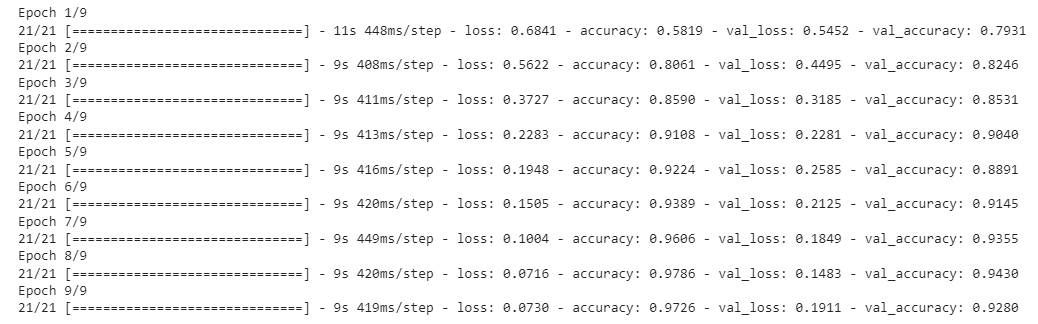
**Результаты в модели VGG21**

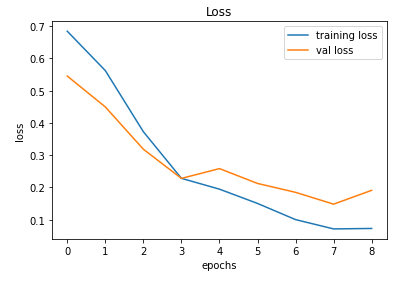
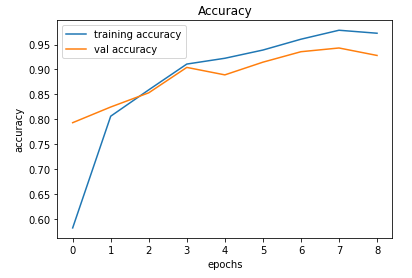
VGG21 — это вариант, который базируется на модели VGG16, которая вкратце состоит из 21 слоев (21 слоев свертки, 13 слоев batch normalization, 5 максимального пулинга, 1 слой flatten, 3 полносвязанных слоя: 2 слоя с 1 слой SoftMax). На Рисунке показана архитектура модели vgg19 в этой системе.

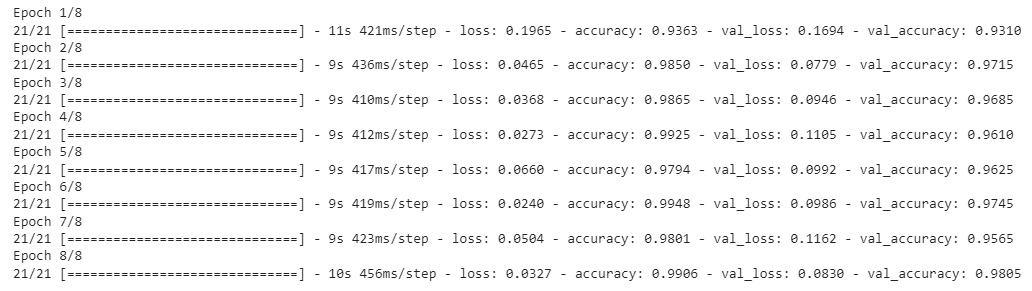
На Рисунке 2.31 показаны результаты модели VGG19 для эпох 8, 9 и 10.











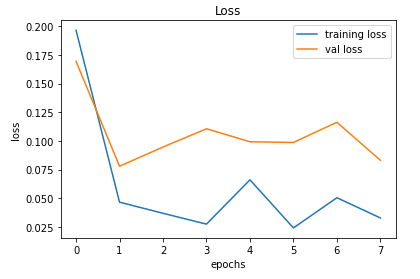
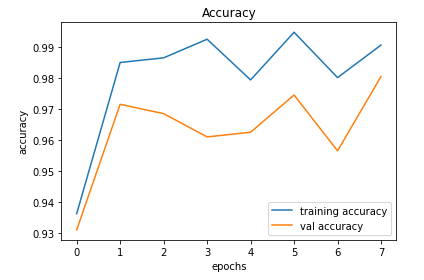


Рисунок 2.31. Показаны результаты модели VGG19 для эпох 8, 9 и 10

Таблица 6. Время, точность и точность проверки VGG19 для эпох 8, 9, 10

|  |  |  |  |
| --- | --- | --- | --- |
| Эпохи | Время | Точности | Точности проверки |
| 10 | 92 | 0,8174 | 0,8246 |
| 9 | 83 | 0,9726 | 0,9280 |
| 8 | 75 | 0,9906 | 0,9805 |

Таблица 7. Потери и потери проверки VGG19 для эпох 8, 9, 10

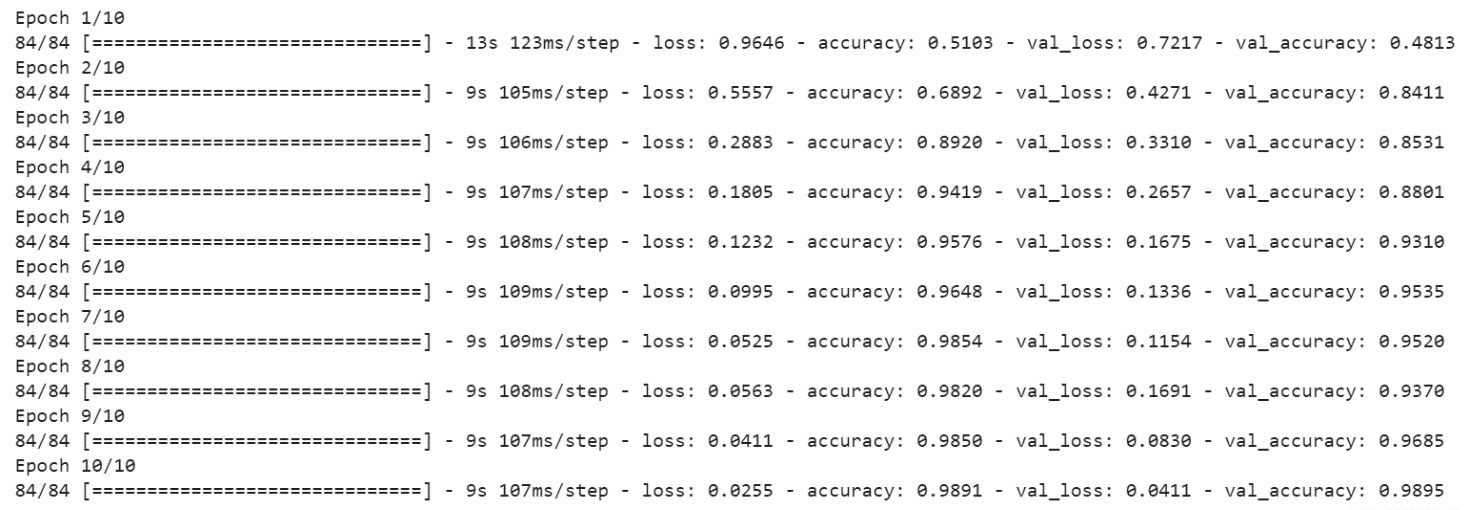
|  |  |  |  |
| --- | --- | --- | --- |
| Эпохи | Время | Потери | Потери проверки |
| 10 | 92 | 0,4275 | 0,4383 |
| 9 | 83 | 0,0730 | 0,1911 |
| 8 | 75 | 0,0327 | 0,0830 |

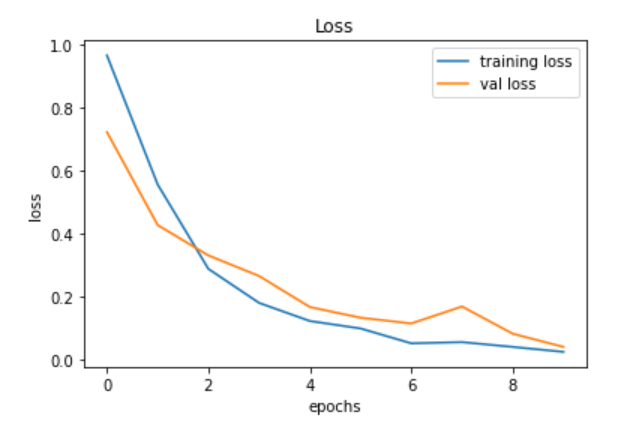
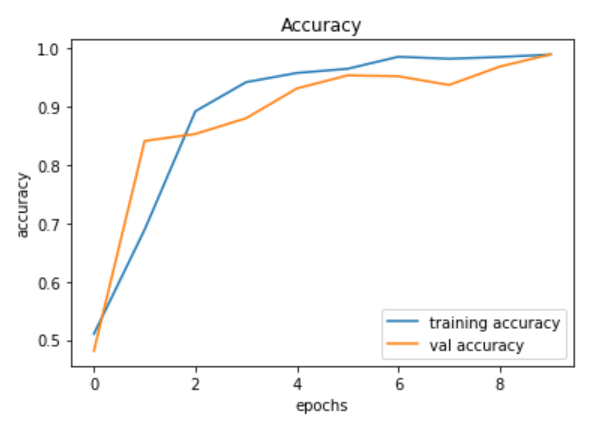
Мы можем видеть точность и потери модели VGG19 в таблицах (6) и (7). По результатам мы должны выбрать эпоху-8.

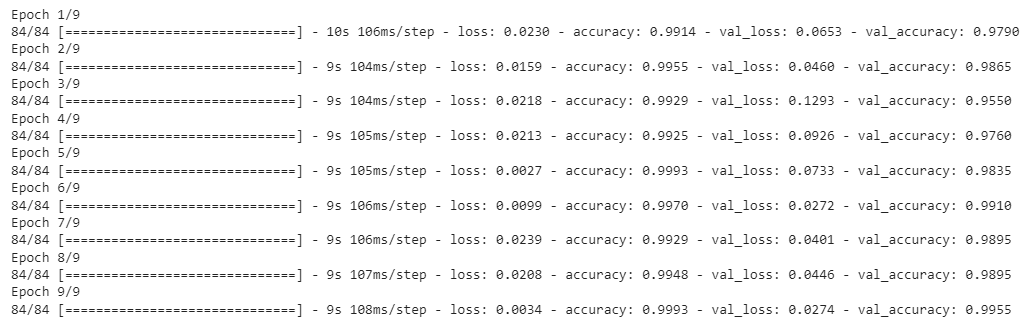
**Результаты в модели VGG16**

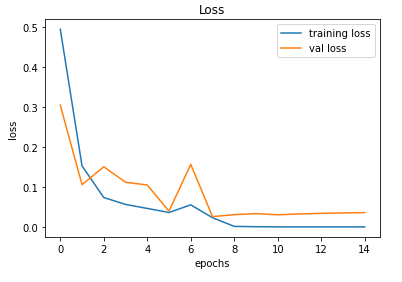
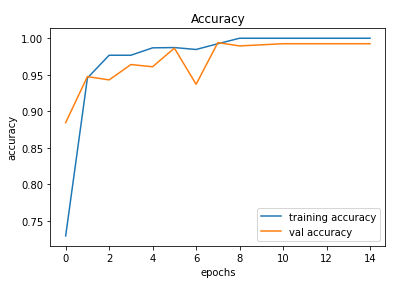
Самая уникальная особенность VGG16 заключается в том, что вместо большого количества гиперпараметров они сосредоточились на наличии слоев свертки фильтра 3x3 с шагом 1 и всегда использовали одно и то же заполнение и слой maxpool фильтра 2x2 с шагом 2. Он последовательно следует этому расположению слоев свертки и максимального пула на протяжении всей архитектуры. В конце он имеет 2 FC (полностью подключенных слоя), за которыми следует softmax для вывода. 16 в VGG16 относится к тому, что он имеет 16 слоев, которые имеют веса. Эта сеть - довольно большая сеть, и она имеет около 138 миллионов (приблизительно) параметров.

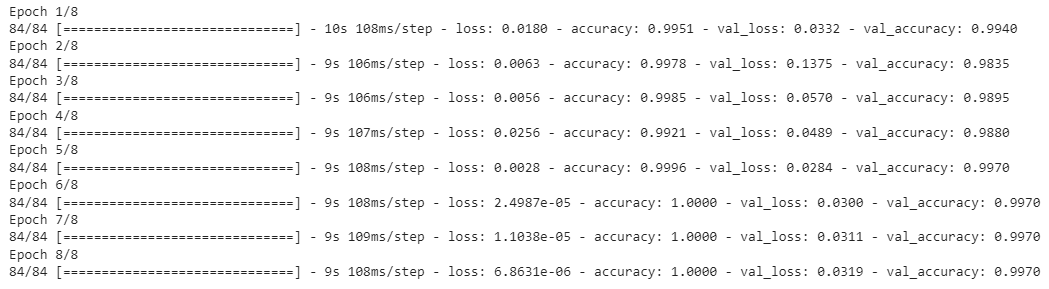
На Рисунке 2.32 показаны результаты модели VGG16 для эпох 8, 9 и 10.











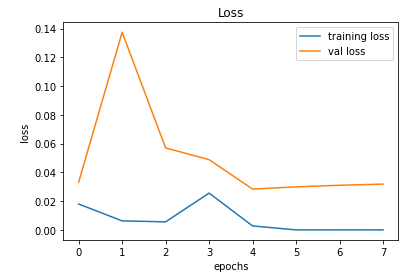
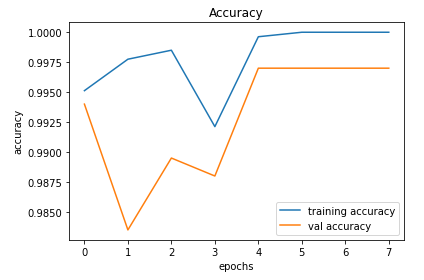


Рисунок 2.32. Показаны результаты модели VGG16 для эпох 8, 9 и 10

Таблица 8. Время, точность и точность проверки VGG16 для эпох 8, 9, 10

|  |  |  |  |
| --- | --- | --- | --- |
| Эпохи | Время | Точности | Точности проверки |
| 10 | 94 | 0,9891 | 0,9895 |
| 9 | 82 | 0,9993 | 0,9955 |
| 8 | 73 | 1,0000 | 0,9970 |

Таблица 9. Потери и потери проверки VGG16 для эпох 8, 9, 10

|  |  |  |  |
| --- | --- | --- | --- |
| Эпохи | Время | Потери | Потери проверки |
| 10 | 94 | 0,0255 | 0,0411 |
| 9 | 82 | 0,0034 | 0,0274 |
| 8 | 73 | 6,8631 | 0,0319 |

По результатам таблица (8) и таблица (9) мы должны выбрать epoch-9.

На рисунках показаны время, потери и точность AlexNet, VGG19 и VGG16. Таким образом, мы видим, что AlexNet - лучшая модель для этой системы.



Рисунок 2.33. Результат тестирования для AlexNet



Рисунок 2.34. Результат тестирования для VGG19



Рисунок 2.35. Результат тестирования для VGG16

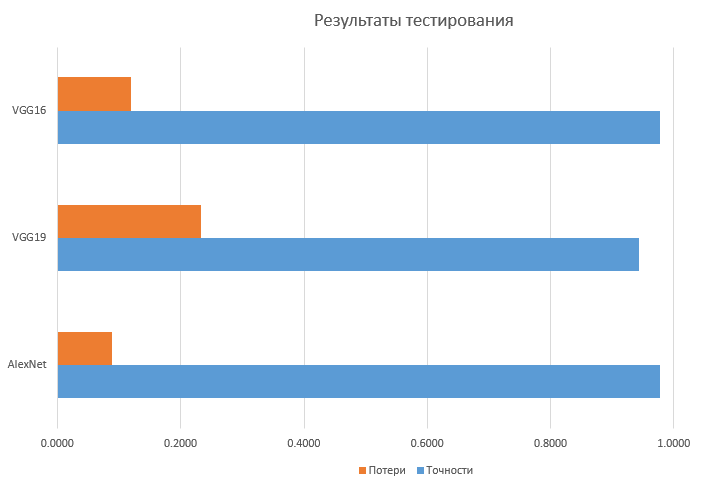


Рисунок 2.39. График результатов тестирования

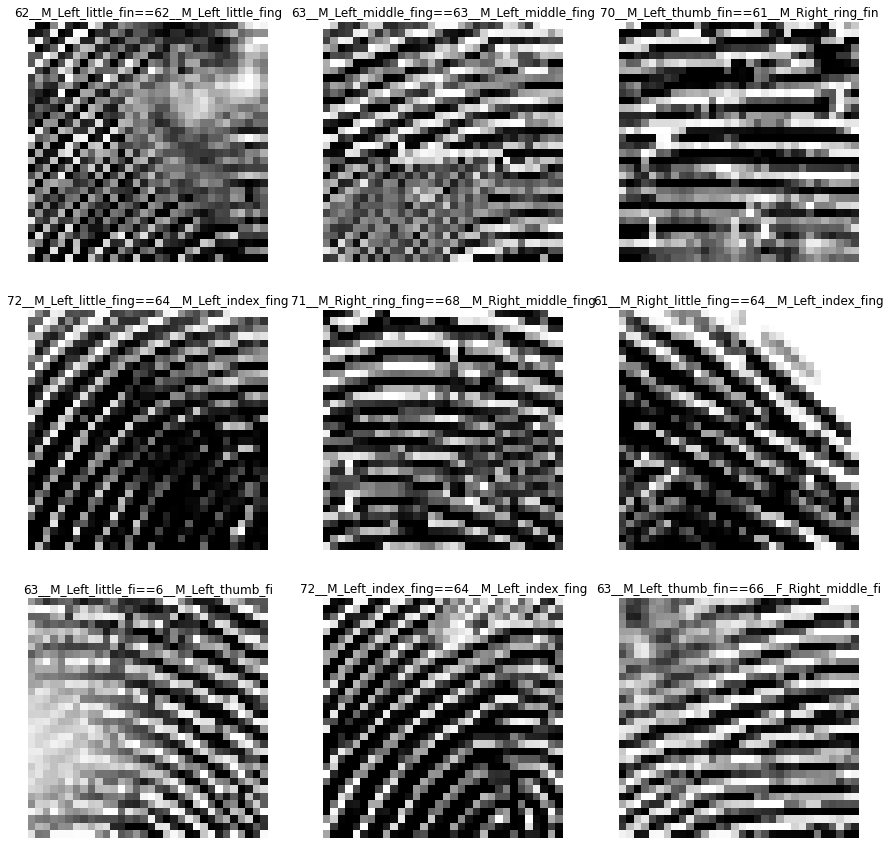


Рисунок 2.40. Выходной результат идентификации отпечатков пальцев

# Заключение

В результате выполнения выпускной работы магистра "Интеллектуальная система идентификации человека по отпечаткам пальцев" были получены следующие основные результаты:

1. Исследована предметная область

2. Изучены методы и подходы для идентификации человека по отпечаткам пальцев

3. Проведён анализ и выбор архитектуры НС

4. Проведён анализ и выбор методов обучения НС

5. Проведён анализ и выбор среды для разработки НС

6. Разработана предварительно нейронная сеть и система для задачи идентификации отпечатков пальцев человека

7. Проведено моделирование разработанной системы, эксперименты и анализ результатов

Во время выполнения выпускной квалификационной работы магистра были использованы научные и инженерные мотоды такие как линейная алгебра, методы математической статистики, методы анализа данных, теория ортимизации, методы матричных вычислений, алгоритмы машинного обучения, процедурное и объектно-ориентированное программирование, технология разработки и сопровождения программ.

# Список использованных источников

1. Афанасьев Г.И., Абулкасимов М.М., Сурикова О.В. Алгоритмы оптимизации, используемые в нейронных сетях, и градиентный спуск // Аспирант и соискатель. 2019. № 6 (114). С.81- 86.

2. Галичий Д.А., Афанасьев Г.И., Нестеров Ю.Г., Распознавание эмоций человека при помощи современных глубокого обучения методов // E-SCIO. 2021. №5. С.317-331.

3. Oh S.K., Lee J.J. , Park C.H., Kim B.S., K.H. Park. New Fingerprint Image Enhancement Using Directional Filter Bank // Journal of WSCG, Vol.11, No.1. — 2003.

4. Pasynkov M. K., Khachay M. Y., Segmentation of fingerprints using convolution neural networks // Proceedings of the 48th International Youth School-Conference "Modern Problems in Mathematics and its Applications" Yekaterinburg, Russia, February 5-11. Vol-1894. 2017. P.215-227.

5. Джейн и Э. Ал, "Введение в биометрическое распознавание", IEEE Tran. О схе-мах и системах для видеотехники, Том 14, №1, стр.: 4-20, 2004.

6. П. Балди и Ю. Шовен, "Нейронные сети для распознавания отпечатков паль-цев", нейронные вычисления, 5, стр. 485-501, 1993.

7. В. Шринивасан и Н. Мурти, "Обнаружение точек сингулярности в изображени-ях отпечатков пальцев", Распознавание образов, 20, стр.513-523., 1992.

8. Д. Майо и Д. Мальтони, "Прямое обнаружение мелочей в серой шкале в отпе-чатках пальцев", IEEE Tran по анализу шаблонов и машинному интеллекту, Том 19, № 1, 1997.

9. S. Al, "Нечеткое управление для выбора признаков отпечатка пальца", Proc. ACCV, том 3, стр.: 767-771, 1995.

10. X. Ся и Л. О'Горман, "Инновации в устройствах захвата отпечатков пальцев", Журнал распознавания образов, Pergamon Press, Том 36, № 2, стр. 361-370, 2002.

11. https://keras.io/api/applications/vgg/

12. https://en.wikipedia.org/wiki/AlexNet

# ПРИЛОЖЕНИЕ А. Техническое задание

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Утверждаю:

\_\_\_\_\_\_\_\_\_\_\_Г.И. Афанасьев

"\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

«Интеллектуальная система идентификации человека по отпечаткам пальцев»

Техническое задание

(вид документа)

Писчая бумага формата А4

(вид носителя)

5

(количество листов)

|  |  |
| --- | --- |
| Исполнитель: | студент группы ИУ5-42М |
| Вей Пхьоу Ту | |
| "\_\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**Москва -2022**

**Содержание**

1. Наименование 120

2. Основание для разработки 120

3. Исполнитель 120

4. Цель разработки 120

5. Задачи, подлежащие решению 120

6. Модули системы 121

6.1 Модуль набора данных отпечатков пальцев 121

6.2 Модуль сопоставления отпечатков пальцев 121

6.3 Нейронная сеть 121

7. Требования к функциям систем 121

8. Требования к оборудованию 121

9. Этапы разработки 122

10. Документация, предъявляемая по окончании работы 122

11. Дополнительные условия 122

**1. Наименование**

Синтез нейронной сети для задачи идентификации по отпечаткам пальцев 100 человек.

**2. Основание для разработки**

Основанием для разработки является задание на выпускную квалификационную работу магистра, подписанное научным руководителем и утверждённое заведующим кафедрой «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана.

**3. Исполнитель**

Исполнителем является студент группы ИУ5-42М МГТУ им. Н.Э. Баумана Вей Пхьоу Ту.

**4. Цель работы**

Целью данной работы является разработка, исследование и оптимизация проектных решений для синтеза системы идентификации отпечатков пальцев человека на основе применения нейронной сети.

**5. Задачи, которые необходимо решению**

* Анализ предметной области
* Исследование моделей нейронных сетей
* Синтез нейронной сети для идентификации отпечатков пальцев
* Тестирование нейронной сети на заданном исходном наборе данных.
* Разработка системы для идентификации различных отпечатков пальцев разных людей.
* Исследование и оптимизация проектных решений

**6. Модули системы**

**6.1 Модуль набора данных отпечатков пальцев**

В этом модуле набор данных состоит из отпечатков пальцев 100 человек, всех их десяти пальцев. В результате получается 1000 уникальных отпечатков пальцев, каждый из которых здесь рассматривается как отдельный образец. Существуют также измененные (синтетические) версии тех же отпечатков пальцев, сгруппированные по 3 уровням сложности (в соответствии с поставщиком набора данных): легкий, средний и сложный.

**6.2 Модуль сопоставления отпечатков пальцев**

Это базовый модуль системы, в котором на основе нейросетях моделей результаты могут быть с хорошей точностью.

**6.3 Нейронная сеть**

Нейронная сеть реализована с помощью моделей сверточных нейронных сетей.

**7. Требования к функциям системы**

- Входное изображение отпечатка пальца должно быть высокого качества, чтобы сканер отпечатков пальцев был лучше других.

- Точность сопоставления отпечатков пальцев составляет не менее 90% с небольшими потерями.

**8. Требования к оборудованию**

1. Компьютер: по крайней мере со следующими характеристиками:

Процессор: Intel (R) Core (TM) i7-1165G7 CPU @ 2,80 ГГц 2,80 ГГц

Оперативная память: 16,00 ГБ

Тип системы: 64-битная операционная система, процессор на базе x64

Операционная система Windows 10

1. Камера: для сканирования отпечатка пальца изображение отпечатка пальца должно быть высокого качества, поэтому камера должна быть хорошей.
2. Монитор: для отображения результатов обработки.



Рисунок A.1 – общая блок-схема оборудования

**9. Этапы разработки**

Таблица А.1 – Этапы разработки

|  |  |  |
| --- | --- | --- |
|  | Наименование этапа и содержание работ | Сроки исполнения |
|  | Разработка и утверждение технического задания | Февраль 2022 г. |
|  | Анализ методов для системы идентификации по отпечаткам пальцев. | Февраль 2022 г. |
|  | Проектирование нейронной сети | Март 2022 г. |
|  | Проектирование и реализация системы на языке Python | Апрель 2022 г. |
|  | Моделирование, исследование и оптимизация системы | Май 2022 г. |
|  | Оформление ВКРМ | Май 2022 г. |
|  | Защиты работы | Июнь 2022 г. |

**10. Документация, предъявляемая по окончании работы**

Расчетно-пояснительная записка

**11. Дополнительные условия**

Данное техническое задание может уточняться в установленном порядке.

# ПРИЛОЖЕНИЕ Б. Графический материал

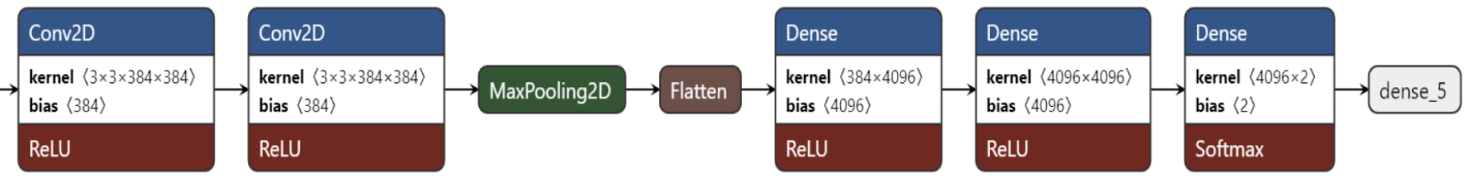
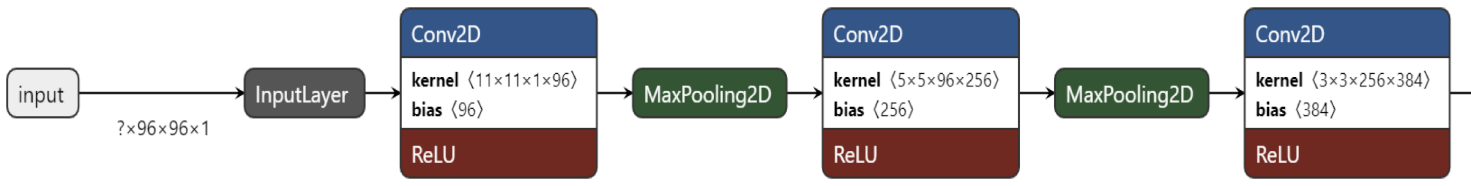
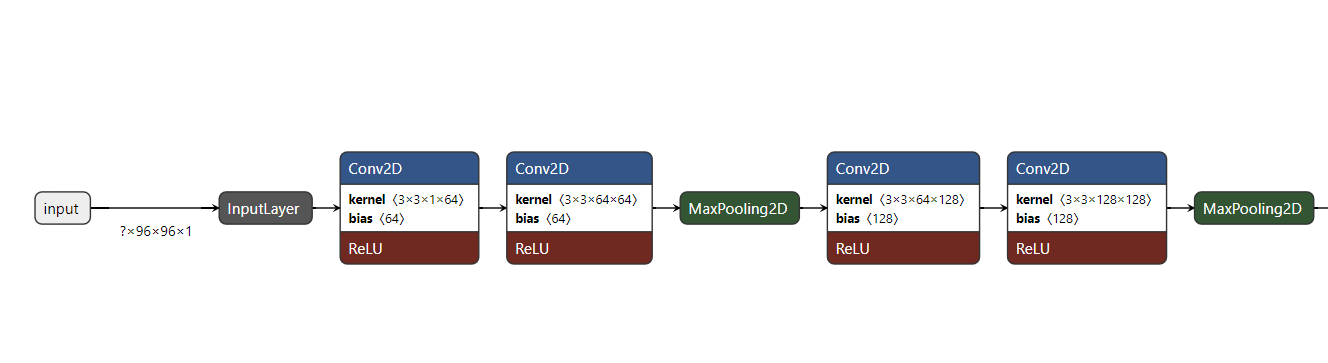
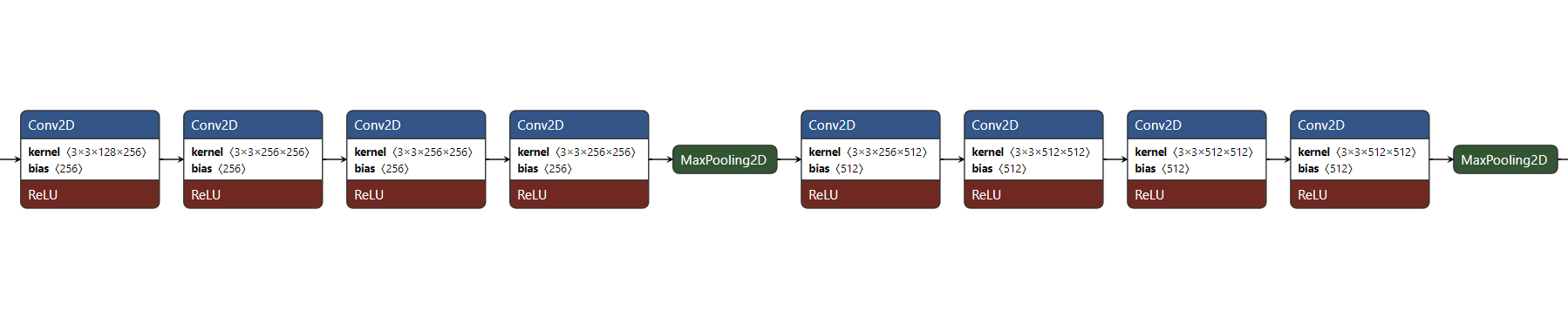


Рисунок Б.1. Архитектура AlexNet





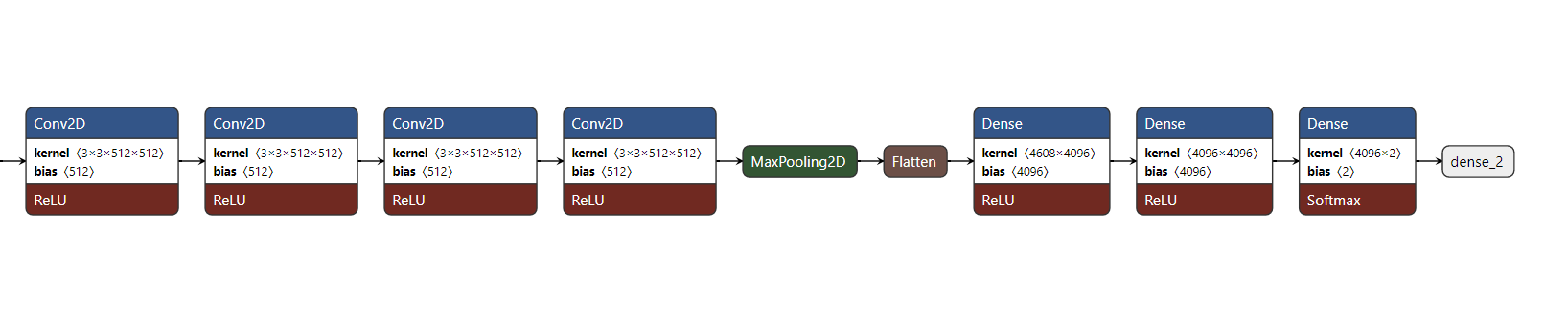
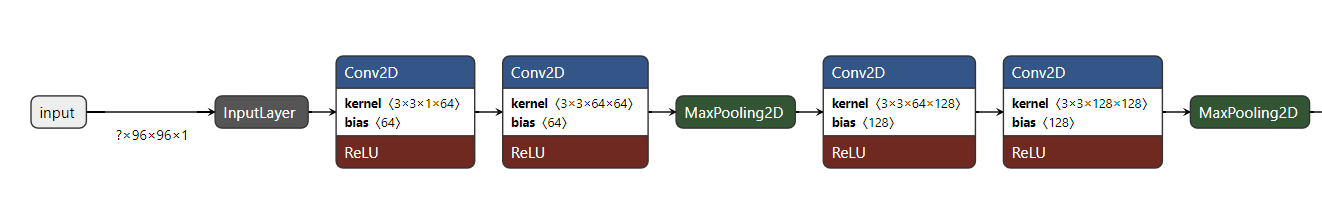
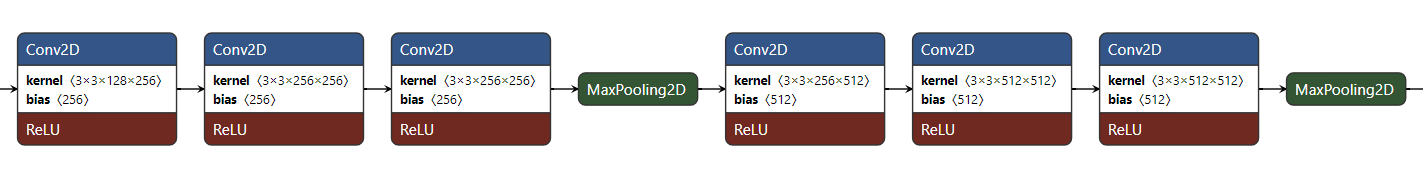


Рисунок Б.2. Архитектура VGG19





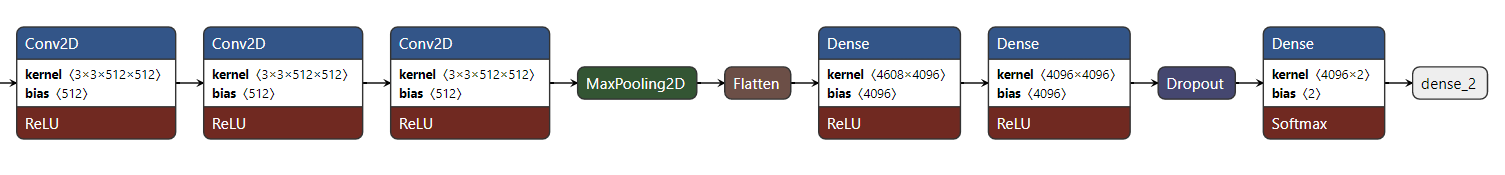


Рисунок Б.3. Архитектура VGG16