МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт радиоэлектроники и информационных технологий

Направление подготовки (специальность) 09.04.01 «Информатика и вычислительная техника»
(код и наименование)

Направленность (профиль) образовательной программы «Теоретическая информатика»
(наименование)

Кафедра «Вычислительные системы и технологии»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

магистра
(бакалавра, магистра, специалиста)

Студента Попов Владислав Антонович (Ф.И.О.) группы М21-ИВТ-3

на тему Модель и алгоритмы распознавания человека по изображению лица
(наименование темы работы)

СТУДЕНТ:
(подпись) Попов В.А. (фамилия, и., о.)
28.06.2023 (дата)

РУКОВОДИТЕЛЬ:
(подпись) Гай В.Е. (фамилия, и., о.)
28.06.2023 (дата)

РЕЦЕНЗЕНТ:
(подпись) Шумилкин К.М. (фамилия, и., о.)
28.06.2023 (дата)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ
(подпись) Жевнерчук Д.В (фамилия, и.о.)
30.06.2023 (дата)

КОНСУЛЬТАНТЫ:
1. По нормоконтролю
(подпись) Клеяев П.С. (фамилия, и., о.)
28.06.2023 (дата)

2. По _____
(подпись) _____ (фамилия, и., о.)
_____ (дата)

3. По _____
(подпись) _____ (фамилия, и., о.)
_____ (дата)

ВКР защищена _____ (дата)

протокол № _____

с оценкой _____

Оглавление

Введение

Актуальность исследования. С каждым годом технологии распознавания лиц развиваются и находят все большее применение в различных сферах жизни. На данный момент уже имеется некоторое количество программ распознавания лиц. Они обеспечивают людям много полезных вещей, например:

1) Услуга распознавания лиц для домофонов, которую предлагает Уфанет, действительно может упростить процесс доступа в подъезды и сэкономить время. Она автоматически распознает жителей и открывает двери без необходимости использования ключей или ожидания, что повышает удобство и безопасность.

2) Функция доступа к телефону по лицу, которая доступна в некоторых моделях телефонов, обеспечивает дополнительный уровень безопасности.

3) Применение распознавания лиц в аэропортах для биометрических паспортов является еще одним полезным применением. Это позволяет подтверждать свою личность сканированitv лица, что может быть полезно при отсутствии паспорта или его утере.

Цель работы. Улучшение точности распознавания лиц на изображении, используя ансамбли нейронных сетей

Задачи работы. Для достижения вышеперечисленной цели были поставлены следующие задачи, а именно:

1) Подготовка набора данных изображений лиц, которые система будет распознавать
2) Исследование существующих алгоритмов распознавания лиц
3) Усовершенствование имеющихся алгоритмов распознавания лиц

Объект исследования. Изображение человеческого лица.

Предмет исследования. Методы и алгоритмы распознавания лиц на изображении

Методы исследования. Просмотр и изучение существующей документации, научных статей, методы математической статистики, нейронных сетей, теории распознавания образов, выявления в ней недостатков, попытка исправить эти недостатки.

Научная новизна. Использование ансамбля нейронных сетей для повышения точности распознавания лиц на изображении.

Практическая значимость результатов магистерской работы. Использование системы для разных человеческих нужд, а именно защита информации, поиск людей, подтверждение личности человека, а также другие сферы, где может понадобится эта система для оптимизации какого-нибудь процесса.

1 Обзор подходов к распознаванию человека по изображению лица

1.1 Задача распознавания человека по изображению лица

Задача по распознанию человека включает несколько этапов

1) Обнаружение. Для обнаружения лица на изображении применяются различные методы компьютерного зрения. Один из популярных подходов - использование каскадов Хаара. Каскады Хаара - это метод обнаружения объектов, основанный на выделении характеристических прямоугольных областей, называемых "фильтрами Хаара". Фильтры Хаара обучаются на большом наборе изображений, чтобы выделять особенности лица, такие как глаза, нос, рот и т.д. Каскады Хаара обеспечивают эффективное обнаружение лиц с высокой точностью и скоростью.

2) Исправление искажений. Возможные искажения в изображении лица, такие как повороты, наклоны и изменения масштаба, могут негативно влиять на точность распознавания. Чтобы справиться с этим, на этом этапе применяются методы исправления искажений. Они могут включать в себя геометрические трансформации, такие как повороты и масштабирование, чтобы привести лицо в стандартизированное положение и размер. Также могут использоваться методы коррекции освещения и контраста для улучшения качества изображения и устранения шума или искажений, вызванных плохим освещением или другими факторами.

3) Идентификация лица по вектору. На последнем этапе векторы признаков, полученные на предыдущем этапе, сравниваются с уже имеющимися данными для идентификации человека. Это может быть выполнено с помощью различных методов сравнения, таких как евклидово расстояние, косинусное расстояние или методы машинного обучения, такие как метод опорных векторов или нейронные сети. Сопоставление вектора признаков с предварительно известными данными позволяет определить, к какому человеку относится данное лицо. Возможно использование пороговых значений или вероятностных моделей для принятия решения о сопоставлении.

1.2 Постановка задачи

Существует множество алгоритмов для распознавания лиц от самых простых и неточных до сложных и точных. На данный момент времени точность распознавания не идеальна для всех алгоритмов. Причинами этого могут служить:

1) Несовершенная архитектура нейронной сети. Выбор архитектуры нейронной сети является важным аспектом при разработке системы распознавания лиц. Слишком сложная

или неподходящая архитектура может привести к недостаточной точности или замедлению обучения. Однако, существует множество исследований и разработок в области архитектур нейронных сетей, которые улучшают точность распознавания лиц. Некоторые популярные архитектуры включают в себя сети с сверточными слоями, такие как VGG, ResNet и MobileNet, которые демонстрируют хорошие результаты в задачах распознавания лиц.

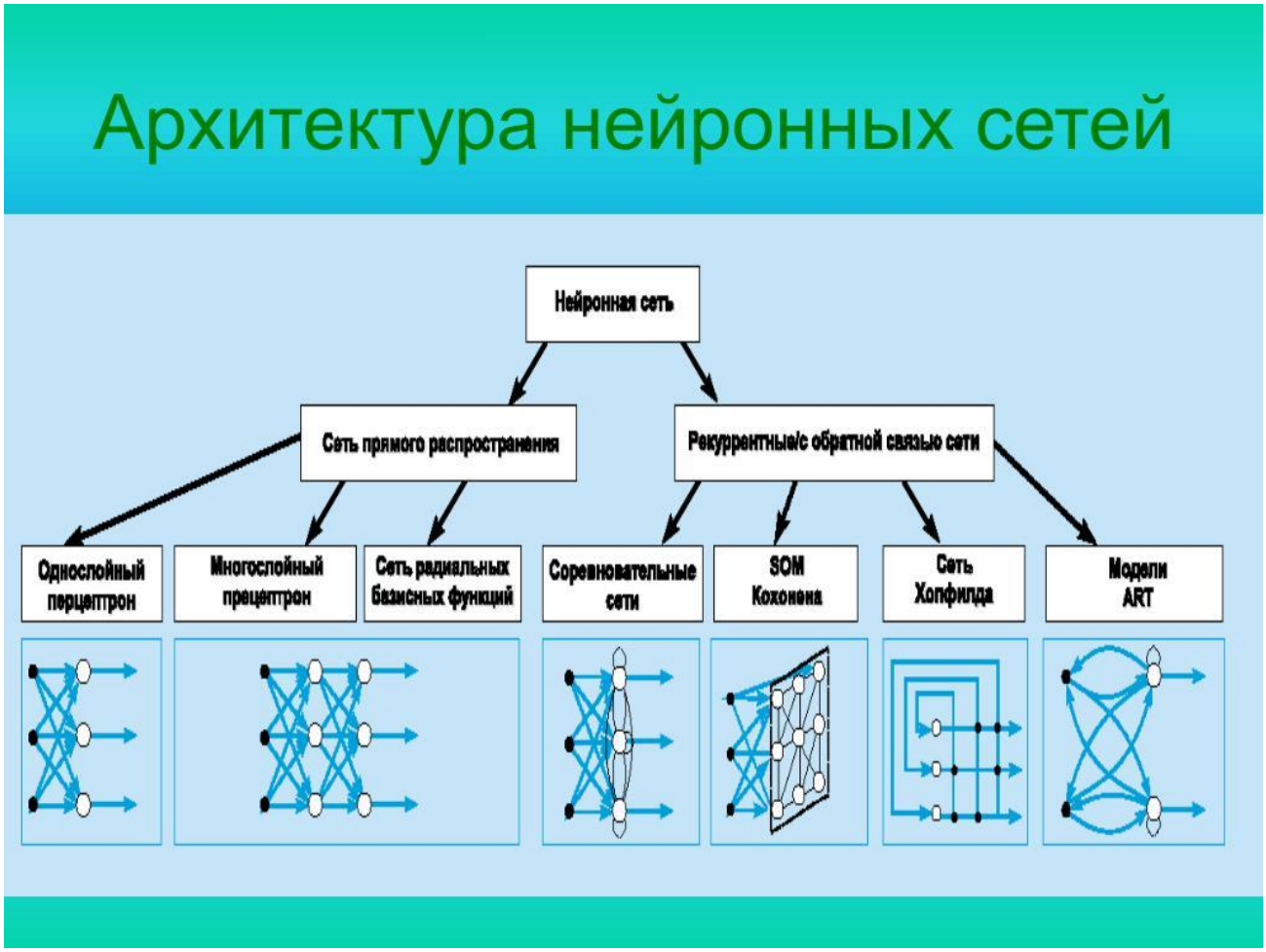

Рисунок 1. – Архитектура нейронной сети

2) Зашумленное изображение. Наличие шума или искажений на изображении может затруднить процесс распознавания лица. Это может быть вызвано различными факторами, такими как низкое качество изображения, неправильная иллюминация, наличие препятствий или объектов на лице и т.д. Для улучшения точности распознавания лиц важно использовать методы предварительной обработки изображения, такие как фильтрация шума, улучшение контрастности, нормализация и т.д. Также можно применять техники обработки изображений, такие как выравнивание лица или устранение искажений, чтобы повысить качество изображения перед процессом распознавания.

Рисунок 2. – Зашумленное изображение

3) Изображение, похожее на лицо человека. Эта проблема называется ложной классификацией или ложным срабатыванием. Во время обнаружения объектов, программа может ошибочно классифицировать некоторые предметы, которые структурно похожи на лица, как реальные лица. Это может произойти, когда на изображении присутствуют предметы или фигуры, имеющие общие черты с лицами, такие как глаза, нос или рот. Решение этой проблемы требует более точных и сложных алгоритмов распознавания, способных различать подобные объекты и делать более точные предсказания.

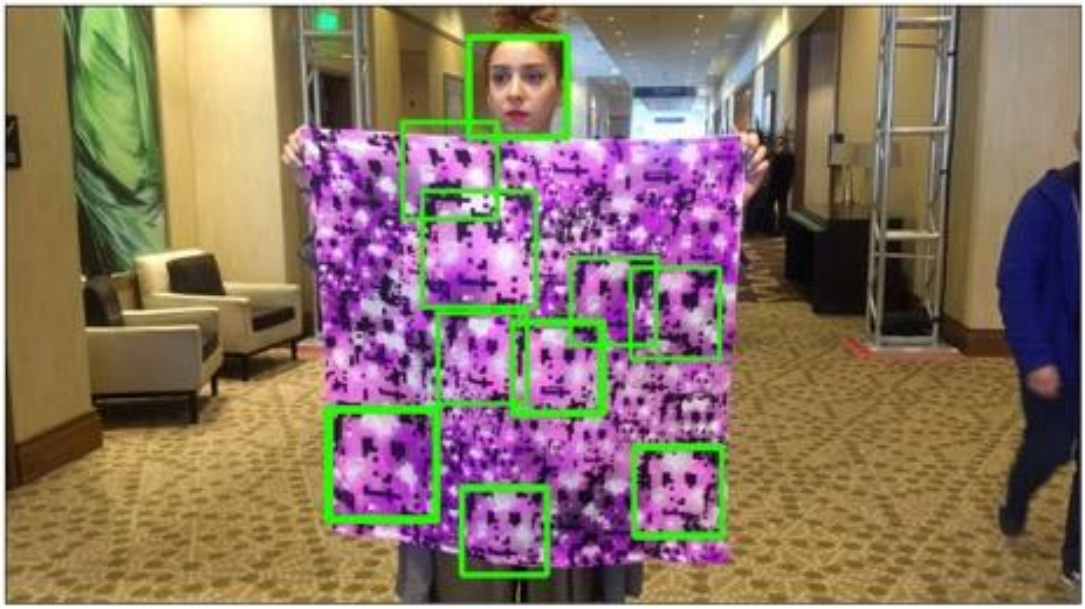

Рисунок 3. – Ложное детектирование

4) Изменение условий съемки: Изменение освещения, угла обзора, фонового шума или других факторов окружающей среды также может повлиять на точность распознавания.

Изображения, сделанные при неблагоприятных условиях, могут содержать недостаточную информацию для надежного распознавания лиц.

1.3 Обзор алгоритмов детектирования лица

1.3.1 Алгоритм R-CNN

R-CNN (Region-based Convolutional Neural Network) - это особый тип сверточной нейронной сети (CNN), который был разработан для решения задач обнаружения, локализации и классификации объектов на изображениях. Основная идея R-CNN состоит в том, чтобы разделить изображение на регионы (предложения областей) и применить сверточную нейронную сеть для извлечения признаков из каждой области. Затем извлеченные признаки передаются в классификатор, который определяет класс объекта, и регрессор, который предсказывает ограничивающие рамки объектов. Классификатор и регрессор могут быть обучены независимо друг от друга. Вывод R-CNN обычно представляет собой набор ограничивающих рамок (bounding boxes), которые плотно соответствуют каждому обнаруженному объекту на изображении. Каждая рамка сопровождается выходным значением класса, указывающим, к какому классу объекта она относится.

Работа R-CNN состоит из следующих шагов:

1) Предложения областей (region proposals): используются алгоритмы, такие как Selective Search, для генерации кандидатов на объекты на изображении. Эти кандидаты представляют собой регионы, которые могут содержать объекты.

2) Извлечение признаков (feature extraction): каждая предложенная область обрабатывается сверточной нейронной сетью, например, предобученной на ImageNet, для извлечения признаков.

3) Классификация и локализация: извлеченные признаки передаются в классификатор и регрессор, которые определяют класс объекта и точно определяют его ограничивающую рамку.

4) Подгонка ограничивающих рамок (bounding box refinement): применяются методы регрессии для точной корректировки предложенных рамок и улучшения их соответствия объектам.

R-CNN была значимым прорывом в области обнаружения объектов и с тех пор была дальше развита, включая улучшенные модели, такие как Fast R-CNN и Faster R-CNN, которые повысили производительность и скорость работы.

Недостатки:

1) Тренировка данных громоздка и слишком длинна

2) Обучение проходит в несколько этапов (например, предложение региона обучения по сравнению с классификатором)

3) Сеть слишком медленная (т.е. при работе с данными, не связанными с обучением)

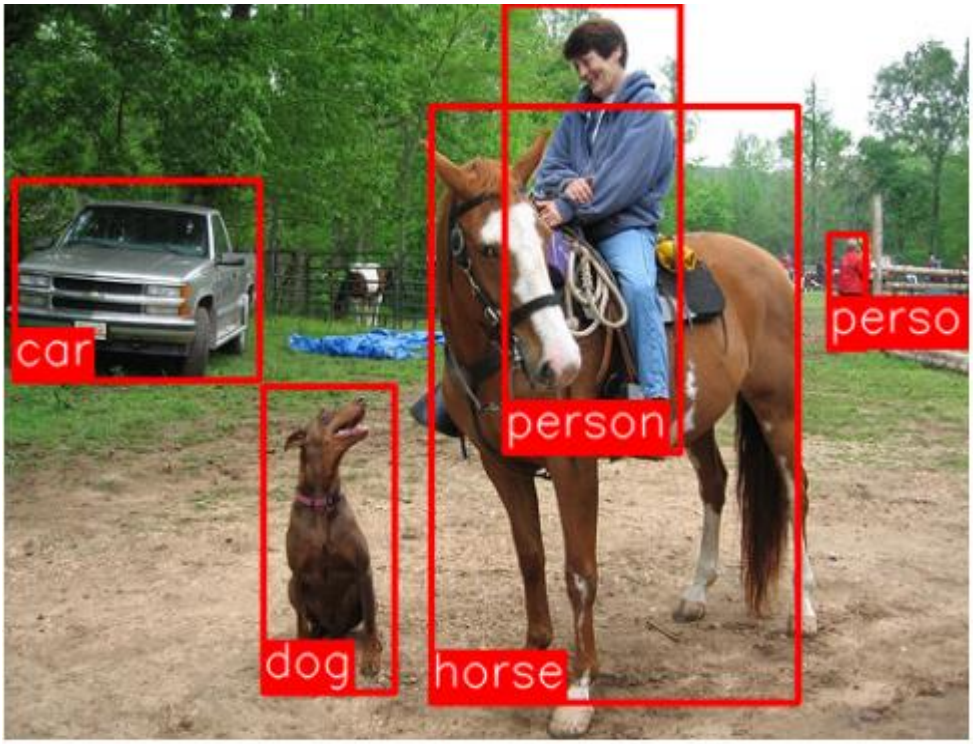

Рисунок 4. – R-CNN

1.3.2 Алгоритм YOLO (You Only Look Once)

YOLO (You Only Look Once) - это алгоритм обнаружения объектов, который использует один проход прямого распространения по сети для делания прогнозов. Он разработан для достижения высокой скорости работы в реальном времени при обнаружении объектов на изображениях или в видеопотоке. Основная идея YOLO заключается в разделении изображения на сетку и применении классификации и локализации для каждой ячейки сетки. Алгоритм предсказывает ограничивающие прямоугольники и оценки достоверности для каждого класса объектов в каждой ячейке. Оценка достоверности отражает уверенность модели в правильности обнаружения объекта. Преимущество YOLO заключается в его способности обнаруживать объекты полностью и точно, так как он рассматривает всё изображение в целом во время обучения и тестирования. Это позволяет алгоритму получать детальную информацию о каждом

объекте и его внешнем виде. Кроме того, YOLO обрабатывает изображение в единичном проходе, что делает его быстрым и эффективным в сравнении с другими подходами, такими как скользящее окно или предложение областей. В последних версиях, таких как YOLOv3, архитектура стала более глубокой и сложной, чтобы повысить точность обнаружения объектов. Однако, из-за увеличения числа слоев, производительность алгоритма может быть немного снижена по сравнению с предыдущими версиями, такими как YOLOv2. Тем не менее, YOLOv3 продемонстрировал улучшение в других аспектах и остается одним из наиболее популярных алгоритмов обнаружения объектов.

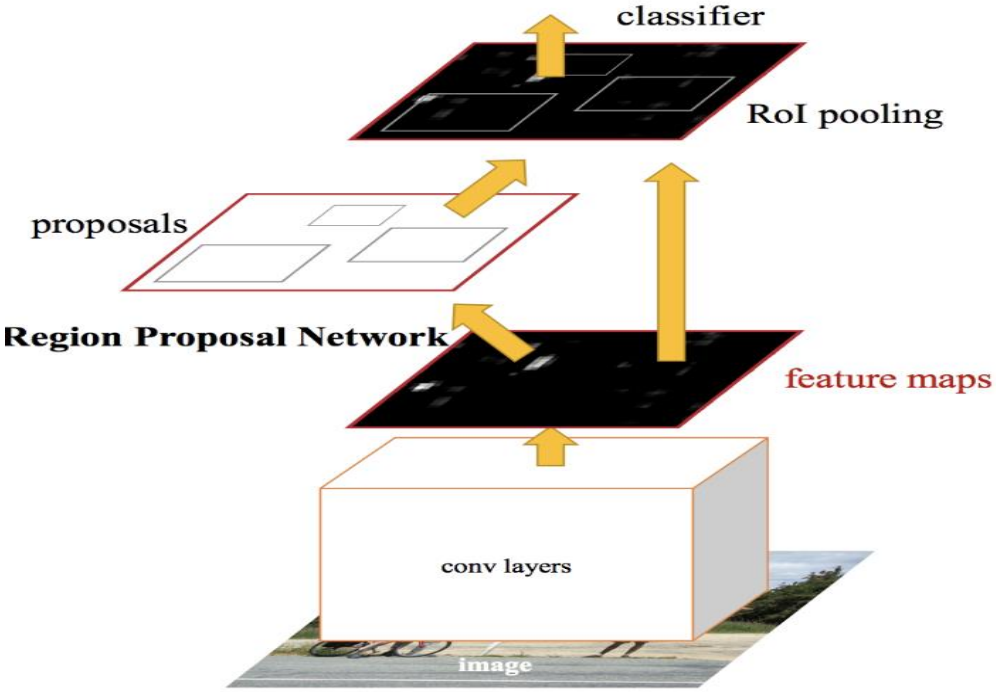

Рисунок 5. – YOLO

1.3.3 SSD (детектор одиночного импульса):

SSD является эффективным и быстрым методом обнаружения объектов, который использует единую нейронную сеть для обнаружения объектов разных масштабов и пропорций. Он не требует предложений по регионам (region proposals) и позволяет сразу предсказывать ограничивающие прямоугольники и классы объектов. Одной из ключевых особенностей SSD является дискретизация выходного пространства ограничивающих прямоугольников. Это означает, что сеть предсказывает ограничивающие прямоугольники в нескольких масштабах и аспектных соотношениях на разных уровнях сверточных слоев. Это позволяет эффективно предсказывать наличие объектов в каждом блоке по умолчанию и обнаруживать объекты разных

размеров. Обучение SSD включает в себя выбор набора ящиков и шкал по умолчанию для обнаружения объектов. Часто используется набор ящиков разных размеров и аспектных соотношений для покрытия различных объектов. Также применяются стратегии жесткого отрицательного анализа (hard negative mining) и увеличения данных (data augmentation) для улучшения качества обучения и предотвращения проблемы несбалансированных классов. SSD демонстрирует высокую скорость обнаружения объектов и хорошую точность результатов, что делает его популярным выбором для многих приложений, таких как автоматическое вождение, видеонаблюдение, обработка изображений и многие другие.

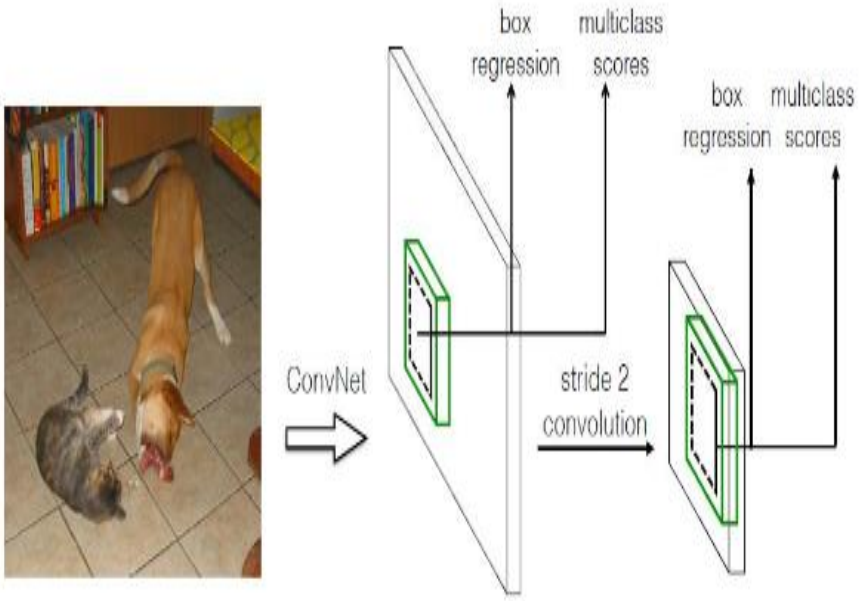

Рисунок 6. – SSD

1.3.4 MTCNN

MTCNN - это метод обнаружения и выравнивания лиц, основанный на глубоком обучении, который позволяет выполнять эти задачи одновременно. В отличие от традиционных методов, MTCNN обеспечивает более высокую производительность и скорость обнаружения лиц. Алгоритм MTCNN состоит из трех подсетей: сети предложений (Proposal Network, P-Net), сети уточнения (Refine Network, R-Net) и сети вывода (Output Network, O-Net) [11]. Каждая подсеть работает на разных уровнях детализации и выполняет определенные задачи.

1) P-Net является первым этапом и обрабатывает входное изображение для грубого обнаружения лиц. Он генерирует предложения (регионы), которые могут содержать лица, и проводит первичную классификацию, чтобы определить наличие лица в каждом предложении.

2) R-Net является вторым этапом и выполняет более точное обнаружение лиц. Он принимает предложения, полученные от P-Net, и уточняет их положение, а также выполняет классификацию для дальнейшей фильтрации ложных срабатываний.

3) O-Net является последним этапом и осуществляет детализированное обнаружение и выравнивание лиц. Данный алгоритм улучшает классификацию лиц, основываясь на дополнительных данных, предоставляемых R-Net. Он также предсказывает координаты ключевых точек на лицах, таких как глаза, нос и рот, с более высокой точностью. Последовательная работа этих трех подсетей позволяет MTCNN обнаруживать лица разных размеров и выравнивать их для дальнейшего анализа и распознавания. Этот метод широко применяется в задачах компьютерного зрения, связанных с обработкой лиц, включая распознавание лиц, автоматическое тэгирование фотографий, эмоциональное анализ и другие.

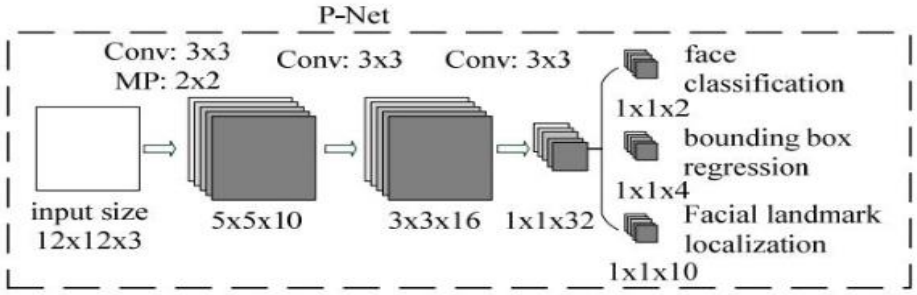

Рисунок 7. – P-Net

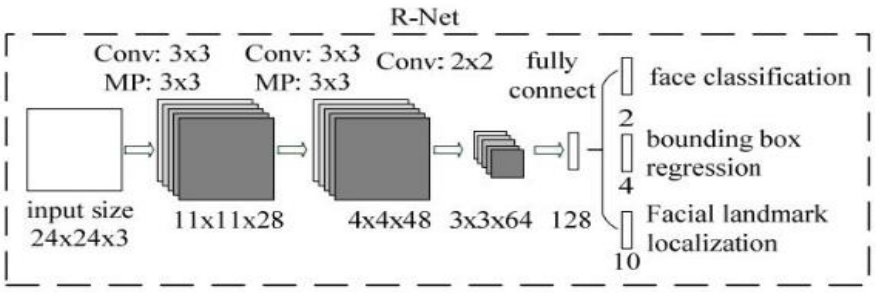

Рисунок 8. – R-Net

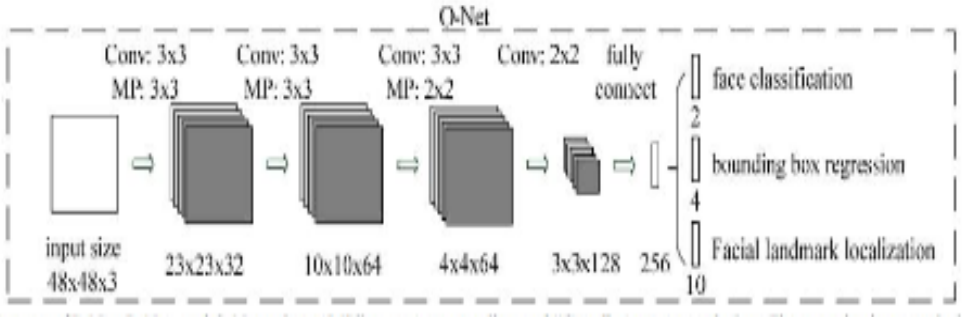

Рисунок 9. – O-Net

1.3.5 Алгоритм Виолы и Джонса

Алгоритм Виолы и Джонса (Viola-Jones) - это один из наиболее известных и широко используемых алгоритмов для обнаружения объектов в изображениях, включая обнаружение лиц. Он основан на комбинации признаков Хаара и использовании классификатора, построенного на основе алгоритма адаптивного бустинга (AdaBoost) [1].

Алгоритм Виолы и Джонса работает в несколько этапов:

1) Подготовка признаков Хаара. Входное изображение разбивается на различные области, и для каждой области вычисляются признаки Хаара. Признаки Хаара представляют собой прямоугольные области разного размера, которые описывают различные характеристики объекта (например, изменение яркости между областями).

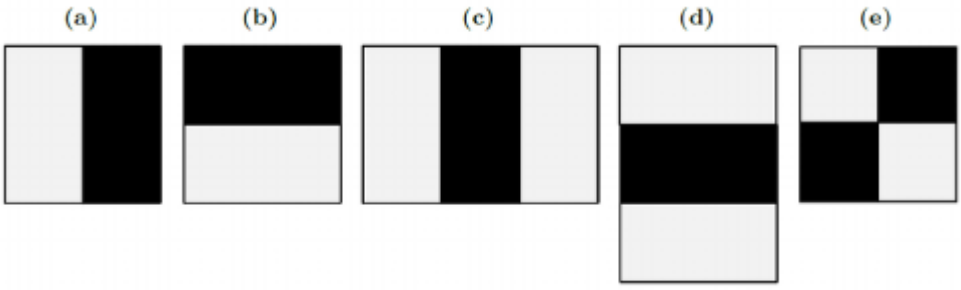

Рисунок 10. – Признаки Хаара

2) Обучение классификатора. С помощью алгоритма адаптивного бустинга (AdaBoost) строится классификатор, который может отличать области изображения, содержащие объект (например, лицо), от областей, не содержащих объект. Классификатор создается путем комбинирования множества простых классификаторов, называемых слабыми классификаторами, которые используются для оценки признаков Хаара.

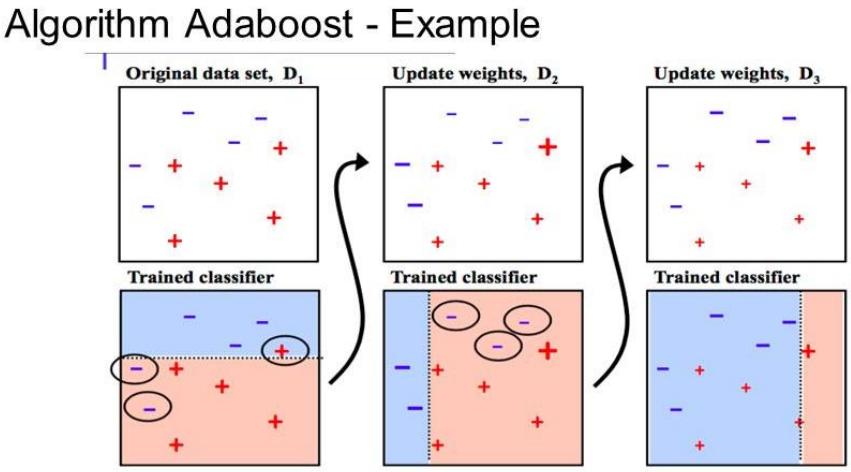

Рисунок 11. – Adaboost

3) Определение регионов объектов. Применяется каскадная структура классификаторов, где каждый классификатор применяется последовательно к различным масштабированным областям изображения. Если область не проходит классификацию на одном из этапов, она отбрасывается. Это позволяет быстро и эффективно отбросить большое количество областей, которые с высокой вероятностью не содержат объект.

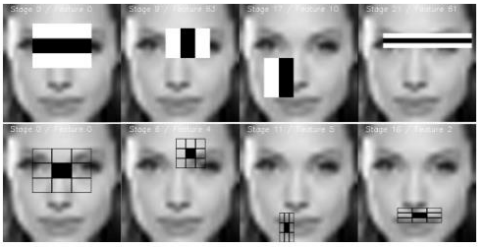

Рисунок 12. – Регионы объектов

4) Отбор лучших регионов и выравнивание: После определения регионов, считающихся объектами, производится их дополнительный анализ для уточнения и выравнивания. Например, можно использовать метод корреляционного выравнивания (сопоставление шаблона) для точного позиционирования объекта.

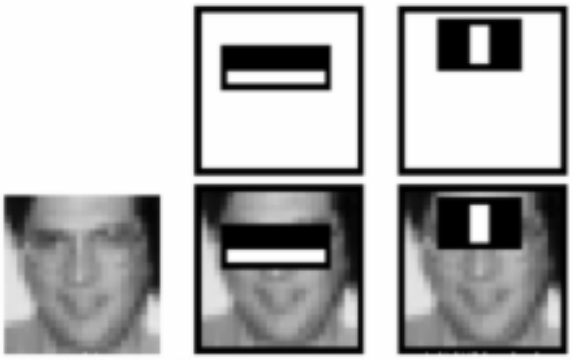

Рисунок 13. – Лучшии регионы

Алгоритм Виолы и Джонса имеет ряд преимуществ, таких как высокая скорость обнаружения и хорошая точность при обнаружении объектов, включая лица. Он широко применяется в таких областях, как автоматическое распознавание лиц, системы видеонаблюдения, дополненная реальность, а также в других задачах компьютерного зрения.

1.3.6 Dlib

Детектор лица в библиотеке Dlib представляет собой эффективный инструмент для обнаружения лиц на изображениях. Он основан на методе градиентного дескриптора ориентированных границ (HOG) и методе опорных векторов (SVM) для классификации лиц и не лиц.

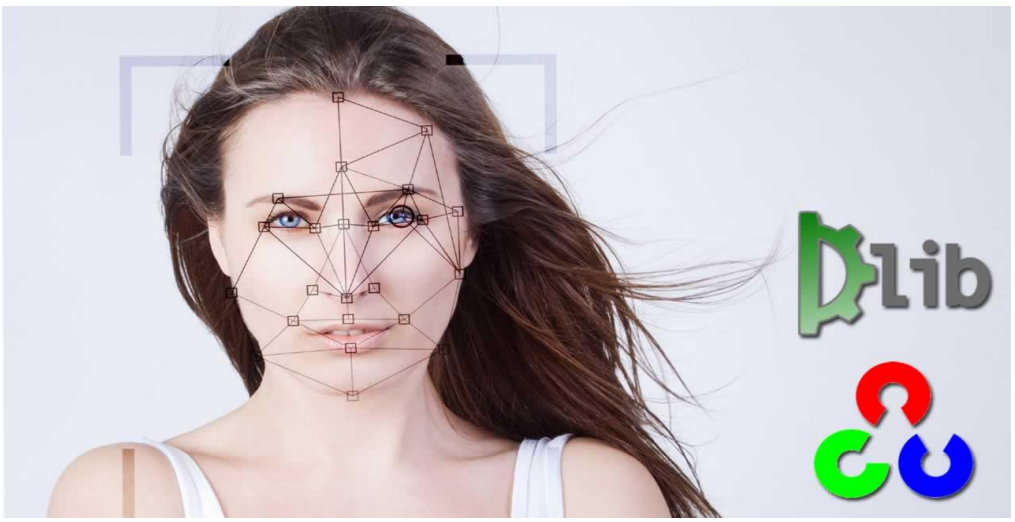

Рисунок 14. – Dlib

Процесс обнаружения лица с использованием Dlib детектора состоит из нескольких этапов. Сначала изображение подвергается предварительной обработке, включающей изменение размера и преобразование в оттенки серого. Затем изображение разбивается на окна разных размеров и форм, которые будут проверяться на наличие лица. Для каждого окна вычисляются градиенты яркости пикселей, которые представляют информацию о направлении и интенсивности изменений яркости. Затем гистограммы направленных градиентов (HOG) создаются для каждого окна, что позволяет представить текстурные особенности лица. Далее, обученный классификатор SVM используется для определения, содержит ли окно лицо или нет. Классификатор обучается на большом наборе изображений с лицами и без них, чтобы научиться различать характерные признаки лиц. После этого применяется нон-максимумное подавление, чтобы удалить дублирующиеся обнаружения и оставить только наиболее вероятные лица. Dlib детектор лица отличается высокой точностью и способностью обнаруживать лица в различных условиях. Он широко применяется в различных приложениях, таких как системы безопасности, автоматическая фокусировка камеры, реализация эффектов и фильтров на изображениях и многие другие. Благодаря своей эффективности и точности, Dlib детектор лица является популярным инструментом для распознавания и обнаружения лиц в различных областях компьютерного зрения и приложениях машинного обучения.

1.3.7 OpenCV DNN

OpenCV DNN (Deep Neural Networks) - это модуль в библиотеке OpenCV, который предоставляет возможности работы с моделями глубокого обучения для различных задач компьютерного зрения. С помощью него вы можете загружать предварительно обученные

модели из популярных фреймворков глубокого обучения, таких как TensorFlow, Caffe, Torch и Darknet. Это позволяет использовать уже обученные модели и применять их для обнаружения объектов, классификации изображений и других задач. Модуль OpenCV DNN обеспечивает удобный интерфейс для загрузки моделей и выполнения вычислений на входных данных. Он может использовать доступные аппаратные ресурсы, включая центральный процессор (CPU) и графический процессор (GPU), для ускорения вычислений и достижения более быстрых результатов. Кроме того, OpenCV DNN предоставляет возможность манипулировать слоями модели, что позволяет настраивать архитектуру сети и изменять ее параметры. Вы также можете применять постобработку к выходным данным модели, например, для применения подавления немаксимума при обнаружении объектов. В целом, OpenCV DNN является мощным инструментом для работы с моделями глубокого обучения в сфере компьютерного зрения.Он предоставляет широкий набор функций и возможностей для загрузки, выполнения и настройки моделей, что делает его полезным инструментом для разработчиков и исследователей, работающих в области распознавания и обработки лиц.

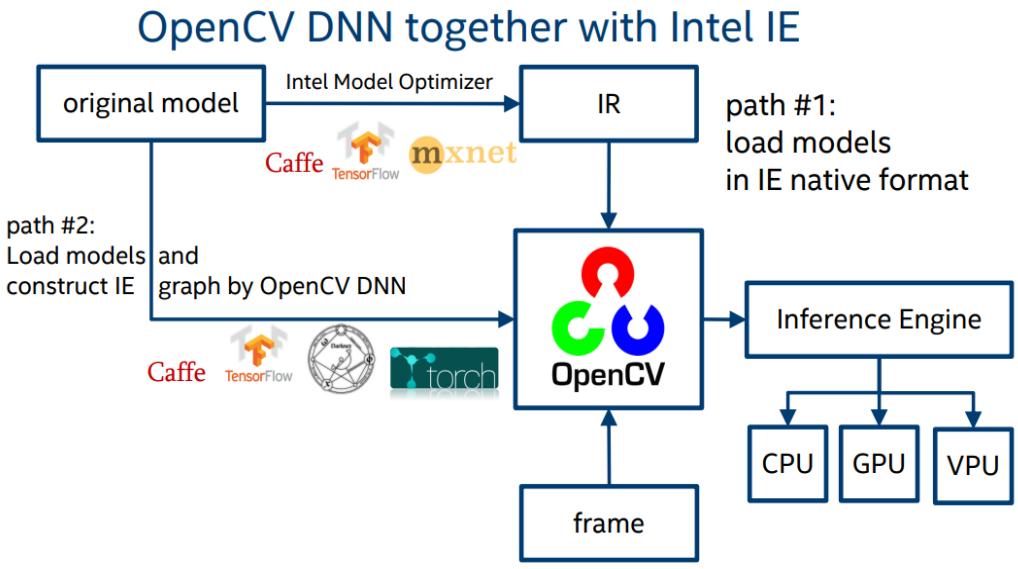

Рисунок 15. – OpenCV DNN

1.3.8 Mediapipe

Mediapipe - это мощная библиотека компьютерного зрения, разработанная Google. Она предоставляет разработчикам широкий спектр инструментов и алгоритмов для работы с визуальными данными, такими как видео и изображения. Одной из ключевых особенностей Mediapipe является его модульная структура, которая позволяет разработчикам комбинировать и настраивать готовые компоненты для выполнения конкретных задач компьютерного зрения.

Библиотека включает в себя компоненты для обнаружения и отслеживания объектов, распознавания лиц, анализа жестов, извлечения структуры скелета и оценки глубины. Mediapipe способен работать в режиме реального времени, обрабатывая видео потоки с высокой производительностью [12].

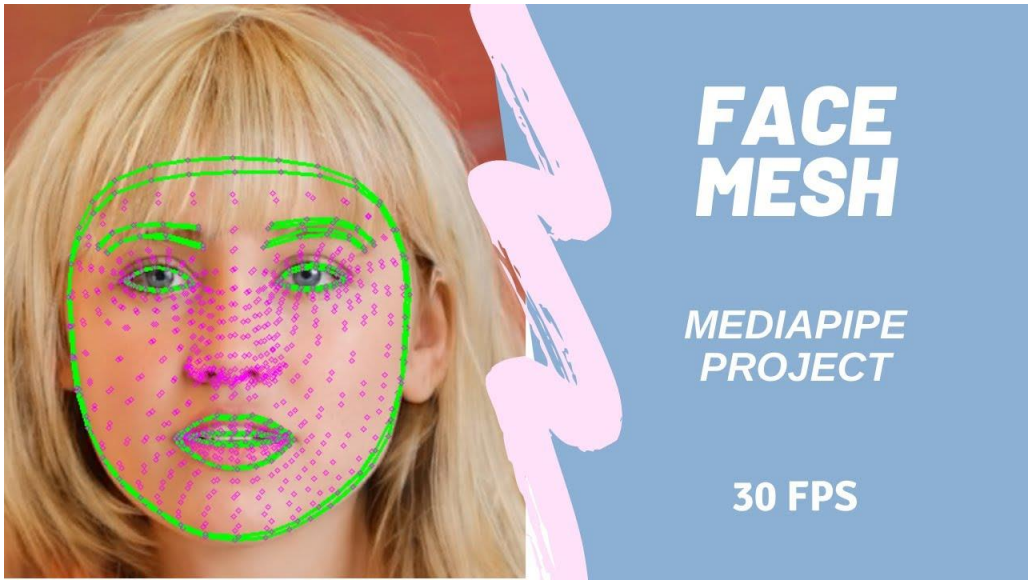

Рисунок 16. – Mediapipe

Он может быть использован на различных платформах, включая настольные компьютеры, мобильные устройства и веб-браузеры. Библиотека оптимизирована для использования аппаратного ускорения, такого как GPU, что позволяет эффективно обрабатывать видео в реальном времени. Mediapipe предоставляет API на нескольких популярных языках программирования, включая C++, Python и Java. Это обеспечивает гибкость и удобство использования для разработчиков с различными предпочтениями. Библиотека также сопровождается обширной документацией, примерами кода и обучающими материалами, которые помогают разработчикам начать работу и освоить возможности Mediapipe. Приложения Mediapipe распространены во многих областях, включая виртуальную и дополненную реальность, интерактивные системы управления жестами, видеоаналитику и другие. Благодаря своей функциональности и гибкости, Mediapipe представляет собой ценный инструмент для разработчиков, интересующихся компьютерным зрением и созданием передовых приложений на его основе.

1.4 Обзор алгоритмов распознавания лица

1.4.1 Метод опорных векторов.

Метод опорных векторов является алгоритмом машинного обучения, который используется для решения задач классификации и регрессии. Основная идея метода опорных векторов заключается в поиске оптимальной гиперплоскости, которая разделяет два класса данных в признаковом пространстве. Гиперплоскость является (N-1)-мерным подпространством в N-мерном пространстве признаков, где N - количество признаков. Для понимания работы SVM, давайте рассмотрим задачу классификации на два класса "лицо" и "не лицо". У нас есть набор изображений, и каждое изображение представлено вектором признаков, который содержит числовые значения, описывающие эту картинку. Задача метода опорных векторов состоит в том, чтобы найти гиперплоскость, которая наилучшим образом разделяет два класса, то есть максимально увеличивает зазор между классами. Вначале, данные подготавливаются и представляются в виде признакового пространства, где каждое изображение представлено вектором признаков. Затем метод опорных векторов строит гиперплоскость, которая максимизирует зазор между классами. Зазор определяется как расстояние от гиперплоскости до ближайших образцов каждого класса, которые называются опорными векторами. Часто данные не могут быть идеально разделены гиперплоскостью. В таких случаях метод опорных векторов может использовать "мягкие" границы классов, позволяя некоторым образцам попадать в зазор или находиться на неправильной стороне границы. Параметр C контролирует компромисс между увеличением зазора и допуском ошибок классификации. Маленькое значение C создает широкий зазор, допуская больше ошибок, а большое значение C приводит к узкому зазору и меньшему количеству ошибок. Чтобы работать с нелинейными задачами классификации, метода опорных векторов использует функцию ядра (kernel function). Функция ядра позволяет эффективно вычислять скалярные произведения в пространстве более высокой размерности без явного перевода данных в это пространство. Линейное ядро применяется, когда данные линейно разделимы, а для более сложных задач можно использовать полиномиальное ядро или радиально-базисное ядро Гаусса (RBF) [7]. Еще одно преимущество метода опорных векторов состоит в его способности обрабатывать выбросы или шумовые точки. Благодаря двум параллельным гиперплоскостям, которые ограничивают зазор и определяют границы классов, SVM становится более устойчивым к выбросам, которые могут находиться близко к границе классов. Цель SVM - найти гиперплоскость, которая максимизирует зазор между классами и минимизирует ошибки классификации на тренировочных данных. Предполагается, что

минимизация ошибок классификации приведет к лучшей обобщающей способности модели и точности классификации для новых данных. Метод опорных векторов является мощным алгоритмом машинного обучения, который широко применяется в различных областях, включая компьютерное зрение, биоинформатику, финансовый анализ и другие. Он позволяет решать сложные задачи классификации и регрессии с хорошей обобщающей способностью и может быть эффективно применен даже к неразделимым и шумным данным.

Недостатки:

1) Метод не распознает шумы и снижает свою эффективность при их наличии
2) Трудоемкость
3) Не имеет общего подхода к выбору ядра, когда практически невозможно разделить классы

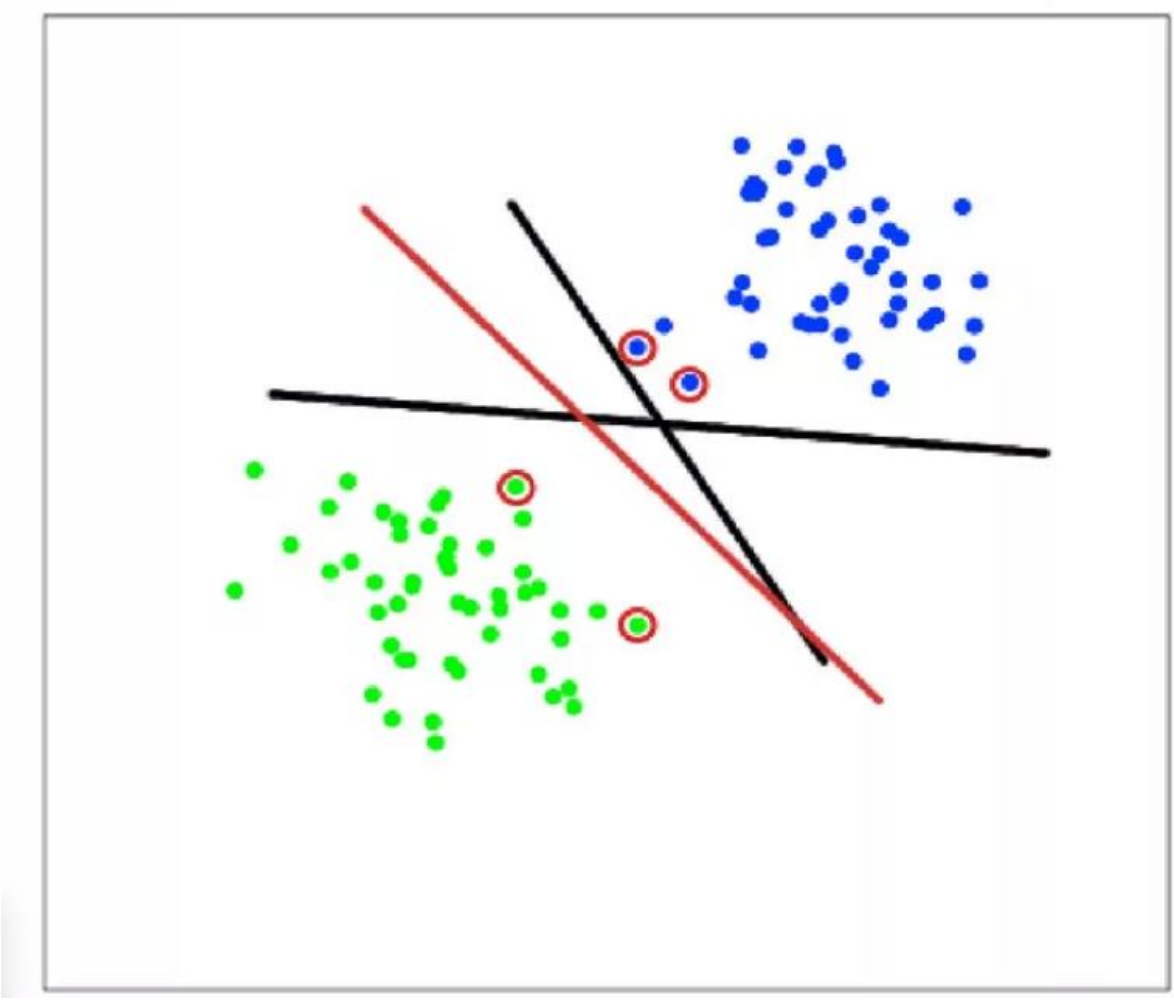

Рисунок 17. – Метод опорных векторов

1.4.2 Многослойные нейронные сети

Многослойные нейронные сети, состоящие из сверточных слоев для извлечения признаков и полносвязанных слоев для классификации, обладают способностью обнаруживать сложные и скрытые зависимости между признаками в данных. Они могут выявлять не только форму объектов, но и более высокоуровневые характеристики, такие как выражение лица, положение объектов, освещение и т.д. Это делает их более точными и способными к более глубокому анализу и интерпретации изображений. Многослойные нейронные сети могут быть использованы для обнаружения объектов на изображениях, классификации изображений по различным признакам (цвет, форма, текстура и т.д.), а также в задачах систем видеонаблюдения, обнаружения лиц, безопасности, медицинской диагностики и других областях. Они позволяют автоматически обучаться на больших наборах данных и применять полученные знания для анализа новых изображений с высокой точностью. Важным аспектом использования многослойных нейронных сетей является правильная архитектура и методы обучения. Разработка оптимальных архитектур сетей, выбор функций активации, оптимизационных алгоритмов и подходящих методов регуляризации являются важными задачами в достижении высокой производительности и точности моделей. В целом, многослойные нейронные сети представляют собой мощный инструмент для анализа изображений, распознавания объектов и классификации, и их использование может привести к достижению высоких результатов в различных приложениях [2].

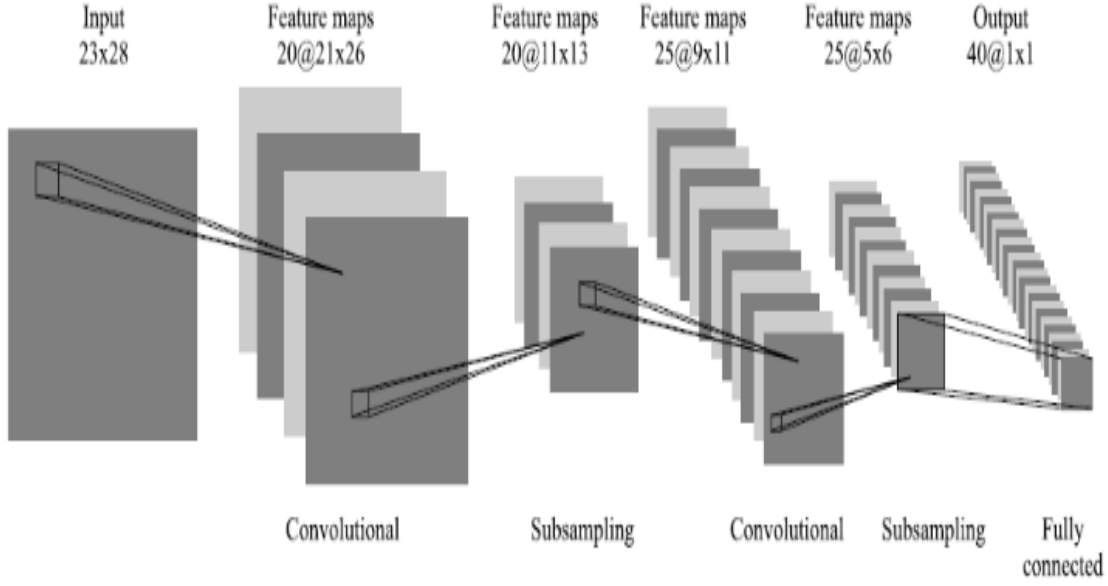

Рисунок 18. – CNN

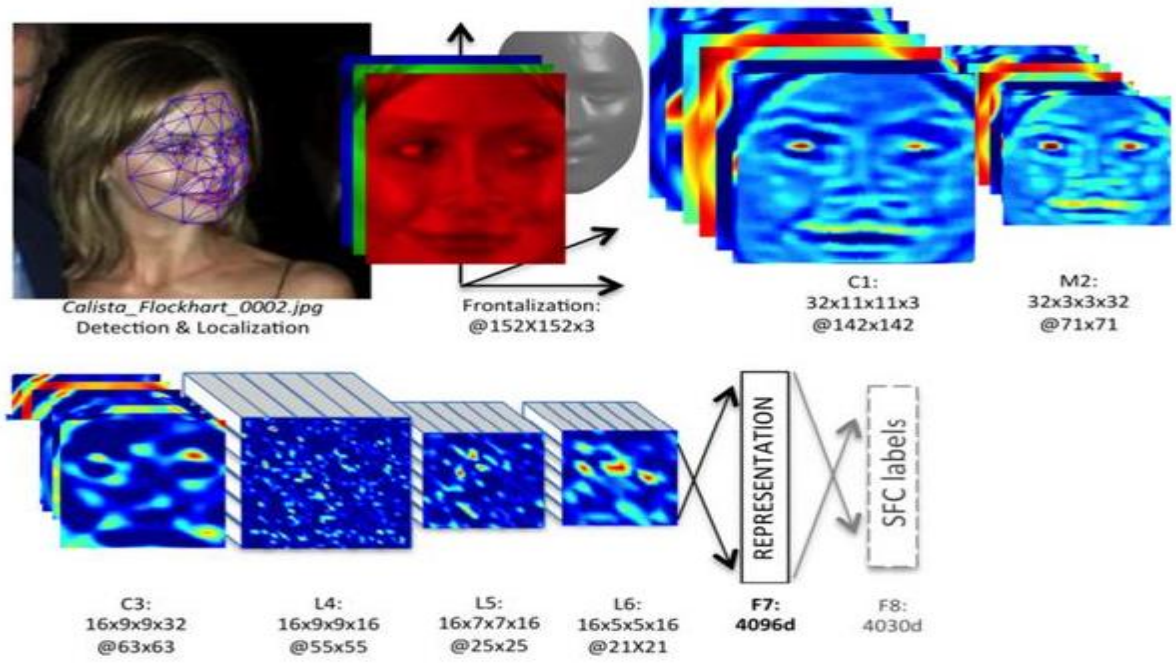

Рисунок 19. – CNN демонстрация

Недостатки:

1) При добавлении новых лиц имеется необходимость полного переобучения сети
2) Проблема самих НС (переобучение, недообучение и т. д.
3) Сложность выбора архитектуры

1.4.3 Скрытые Марковские модели

Марковские модели являются стохастическими моделями, применяемыми для описания различных процессов, где считается, что текущее состояние зависит только от предыдущего состояния. В случае Марковских моделей первого порядка каждое состояние имеет только одно последующее состояние, и каждый наблюдаемый символ генерируется при переходе в новое состояние. В области распознавания речи широко применяются линейные Марковские модели, которые учитывают временные характеристики речевого сигнала. Они учитывают порядок следования сегментов сигнала, их взаимное расположение, а также возможность локальных растяжений или сжатий. Аналогично, двумерные Марковские модели применяются в распознавании изображений и учитывают искажения изображений и взаимное расположение сегментов не только по горизонтали или вертикали, но и в обоих направлениях одновременно [4]. Псевдодвумерные скрытые Марковские модели являются модификацией двумерных Марковских моделей. Они используют несколько линейных вертикальных моделей нижнего уровня и одну линейную горизонтальную модель верхнего уровня. Модели нижнего уровня не

связаны между собой, и каждое состояние модели верхнего уровня включает в себя последовательность состояний соответствующей модели нижнего уровня. Это позволяет моделировать локальные деформации и взаимное расположение сегментов изображений. Начальная инициализация модели важна для скрытых Марковских моделей. В случае использования всех изображений из тренировочного набора в качестве начальной инициализации всех моделей, каждая модель настраивается на свои изображения. Это помогает моделям учиться характерным деформациям и сопоставлять правильные участки изображений. Использование Марковских моделей, включая линейные и двумерные Марковские модели, позволяет моделировать пространственно-временные зависимости в сигналах и изображениях, что делает их мощным инструментом в области распознавания образов и анализа данных.

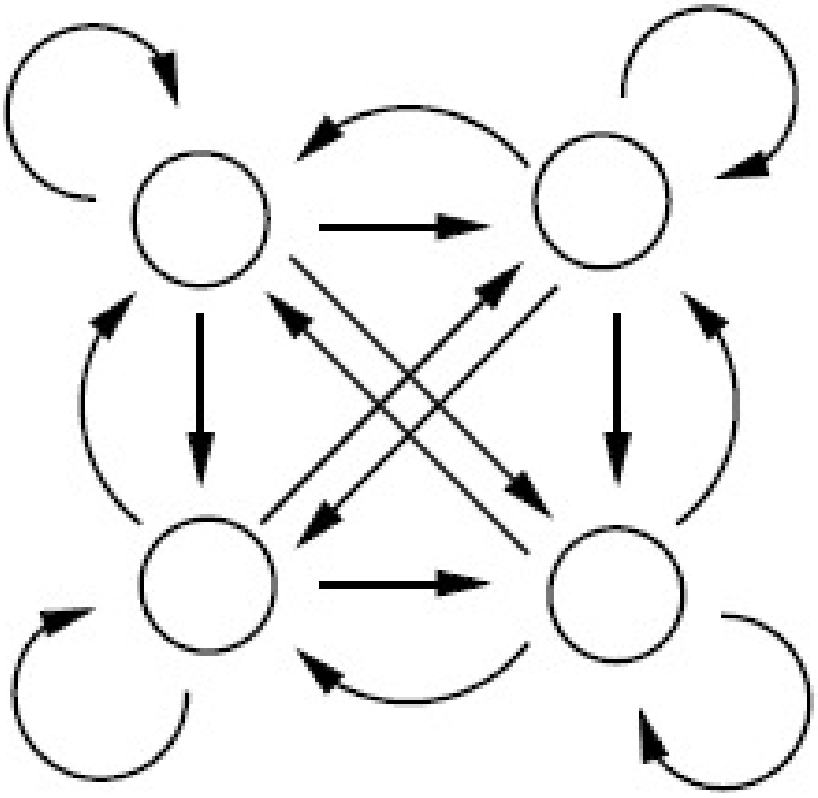

Рисунок 20. – Марковская модель

Недостатки:

1) Необходимость подбора параметров модели для каждой БД
2) Работает локально (в своей) модели, не минимизирует влияние на другие модели

1.4.4 Метод главных компонентов

Метод главных компонент (PCA) является мощным инструментом для снижения размерности данных и извлечения наиболее информативных признаков. В контексте распознавания лиц, PCA может использоваться для представления изображений лиц вектором малой размерности, называемым "главными компонентами" или "собственными лицами" (eigenfaces). Применение PCA для анализа изображений лиц основано на представлении каждого изображения лица как точки в пространстве высокой размерности. PCA ищет основные направления в этом пространстве, вдоль которых данные наиболее изменчивы. Эти основные направления соответствуют главным компонентам, которые могут быть использованы для описания вариаций в данных. Главная идея состоит в том, чтобы найти такое подпространство низкой размерности, в котором большая часть вариации данных сохраняется. В методе главных компонент изображения лиц образуют обучающую выборку, на которой происходит вычисление собственных векторов и собственных значений.

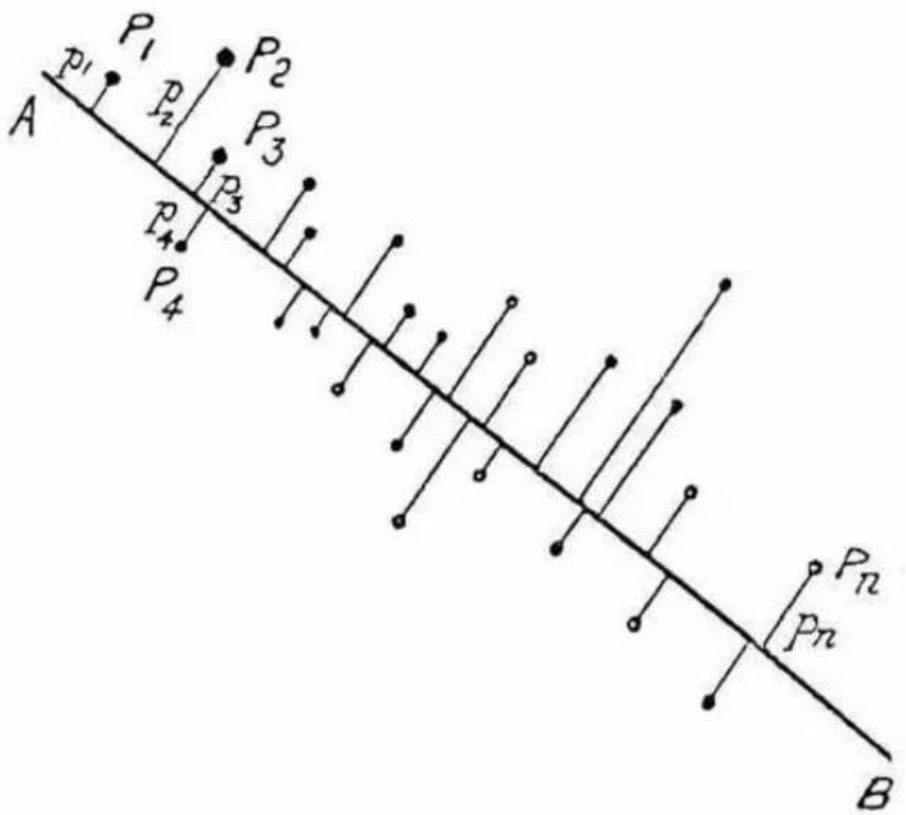

Рисунок 21. – Метод главных компонентов

Собственные векторы представляют собой базисные векторы в пространстве изображений лиц, а собственные значения указывают на важность каждого собственного вектора

в описании вариации данных. Выбираются первые N собственных векторов с наибольшими собственными значениями, которые образуют базисное пространство низкой размерности [5]. Для представления нового изображения лица в этом базисном пространстве, производится проекция изображения на собственные векторы, и получаются коэффициенты, которые служат его сжатым представлением. Коэффициенты могут быть использованы для сравнения и распознавания лиц, а также для поиска в базе данных лиц. Преимущества PCA включают сжатие данных, эффективное представление и распознавание лиц, а также устойчивость к шуму и вариациям. Однако PCA имеет и ограничения. Например, он может не всегда сохранять важные детали, такие как текстура или особенности, которые могут быть важны для определенных приложений. Кроме того, PCA не учитывает нелинейные зависимости между признаками и может быть ограничен

1.4.5 Линейный дискриминантный анализ

Линейный дискриминантный анализ (ЛДА) - это метод, который ищет линейную комбинацию признаков, наилучшим образом разделяющую классы объектов или событий. Он основан на поиске проекции пространства изображений на пространство признаков, которая одновременно минимизирует внутриклассовую дисперсию и максимизирует межклассовое расстояние в пространстве признаков. Цель ЛДА заключается в поиске линейного преобразования признаков, которое максимизирует отношение межклассовой дисперсии к внутриклассовой дисперсии. Это позволяет эффективно разделять различные классы объектов. Важно выбирать такую проекцию на пространство признаков, которая максимизирует разделение между классами. Это особенно важно, чтобы избежать смешивания классов и обеспечить успешное распознавание объектов.

Таким образом, ЛДА находит оптимальную проекцию пространства изображений на пространство признаков, чтобы достичь наилучшего разделения между классами и обеспечить эффективное распознавание объектов. Процесс линейного дискриминантного анализа включает следующие шаги:

1) Подготовка обучающей выборки, включающей объекты из различных классов.
2) Вычисление средних значений признаков для каждого класса.
3) Вычисление внутриклассовой матрицы рассеяния, которая измеряет вариацию признаков внутри каждого класса.

4) Вычисление межклассовой матрицы рассеяния, которая измеряет различие между классами.

5) Решение обобщенной задачи на собственные значения и собственные векторы для комбинирования признаков.

6) Выбор лучших комбинаций признаков на основе критериев межклассовой и внутриклассовой дисперсии.

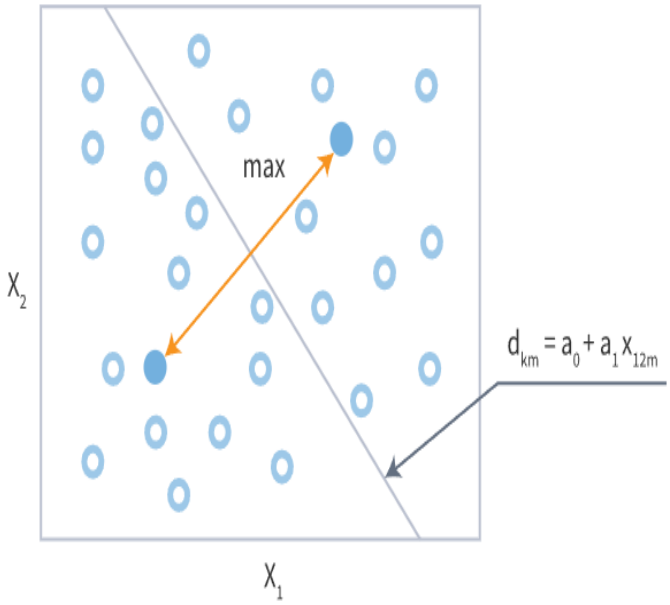

Рисунок 22. – Линейный дискретный анализ

Полученные линейные комбинации признаков могут быть использованы для классификации новых объектов или сокращения размерности пространства признаков перед последующей классификацией. ЛДА широко применяется в области распознавания образов, биометрии, компьютерного зрения и других задачах классификации и анализа данных, особенно когда имеется ограниченное количество обучающих данных.

1.4.6 Алгоритм Local Binary Patterns Histograms

Алгоритм Local Binary Patterns Histograms (LBPH) представляет собой простой подход, который отмечает пиксели изображения с использованием порогового значения, определенного для окрестности каждого пикселя. Иными словами, LBPH анализирует локальную структуру изображения, сравнивая каждый пиксель с его соседями, и преобразует результат в двоичное число. Впервые этот алгоритм был представлен в 1994 году (LBP), и с тех пор он широко признан мощным инструментом для классификации текстур. Основной фокус этого алгоритма заключается в том, чтобы не рассматривать всё изображение как многомерный вектор, а

сосредоточиться только на локальных особенностях объекта. Таким образом, LBPH извлекает и анализирует локальные особенности изображений, что делает его эффективным для задач классификации [3].

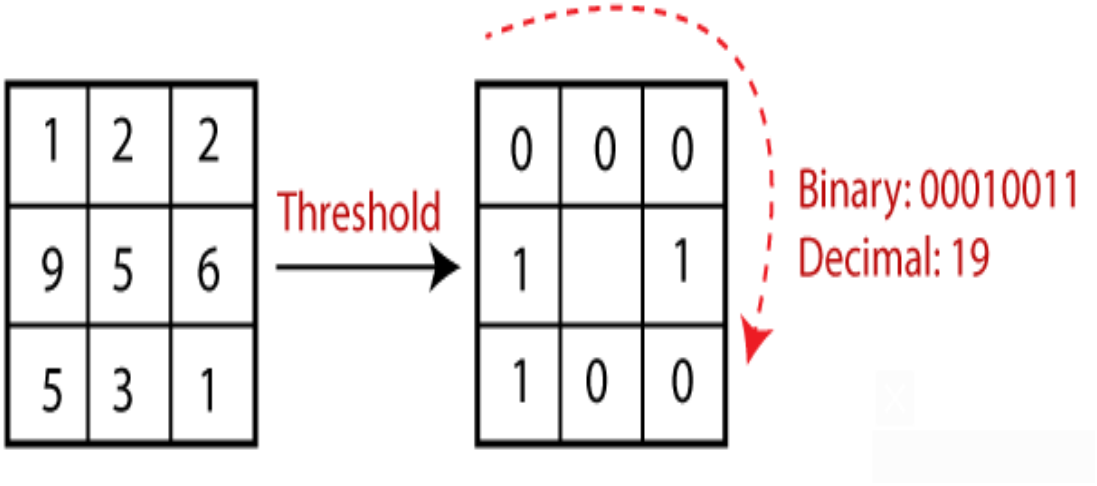

Рисунок 23. – Local BInary Pattern

На изображении выше надо взять пиксель в качестве центра и поставьте порог его соседа. Если интенсивность центрального пикселя больше-равна его соседу, то обозначить его 1, а если нет, то обозначить 0

LBPH принимает четыре параметра:

1) Radius: представляет собой радиус вокруг центрального пикселя. Обычно устанавливается равным 1. Он используется для построения кругового локального бинарного шаблона.

2) Neighbors: количество точек выборки для построения кругового бинарного шаблона.

3) Grid X: количество ячеек в горизонтальном направлении. Чем больше ячеек и тоньше сетка, тем выше размерность результирующего вектора признаков.

4) Grid Y: количество ячеек по вертикали. Чем больше ячеек и тоньше сетка, тем выше размерность результирующего вектора признаков.

Алгоритм начинает с этапа обучения, для которого требуется набор данных, содержащий изображения лиц людей, которых мы хотим распознать. Каждому изображению присваивается уникальный идентификатор, который может быть номером или именем человека. Затем алгоритм использует эту информацию для распознавания входного изображения и предоставления соответствующего результата. Изображение конкретного человека должно иметь тот же идентификатор. На следующем этапе происходит вычисление Local Binary Patterns (LBP) для создания промежуточного изображения, которое описывает исходное изображение и

выделяет характеристики лица. Концепция скользящего окна использует параметры радиуса и соседей для вычисления LBP.

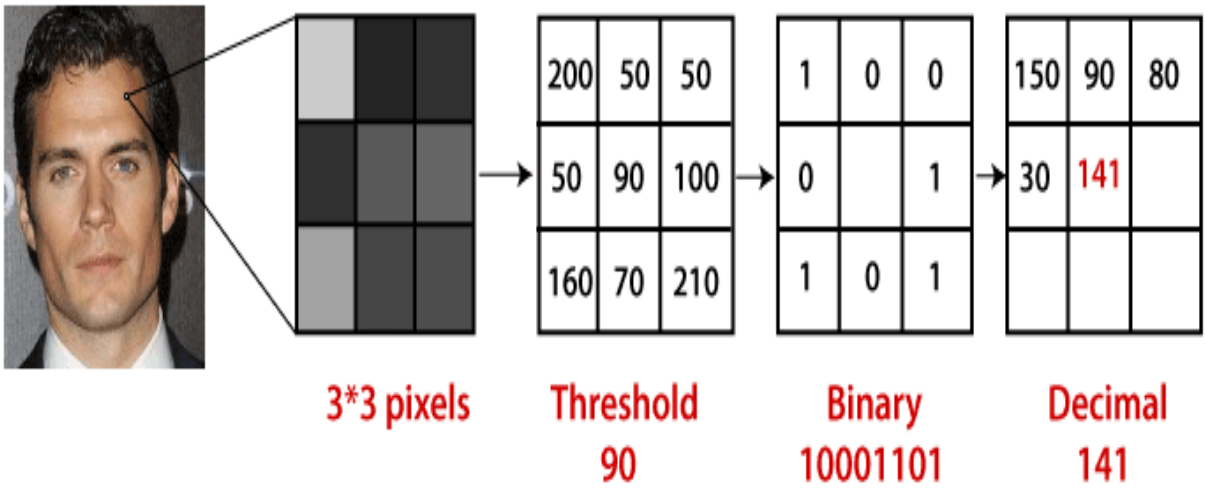

Рисунок 24. – LBPH

Теперь алгоритм хорошо обучен. Извлеченная гистограмма используется для представления каждого изображения из набора обучающих данных. Для нового изображения мы снова выполняем шаги и создаем новую гистограмму. Чтобы найти изображение, соответствующее заданному изображению, нам просто нужно сопоставить две гистограммы и вернуть изображение с ближайшей гистограммой. Существуют различные подходы к сравнению гистограмм (вычислить расстояние между двумя гистограммами), например: евклидово расстояние, хи-квадрат, абсолютное значение и т.д.

Алгоритм вернет идентификатор в качестве вывода изображения с ближайшей гистограммой. Алгоритм также должен возвращать рассчитанное расстояние, которое можно назвать измерением достоверности. Если достоверность ниже порогового значения, это означает, что алгоритм успешно распознал лицо.

1.4.7 Сиамские сети

С помощью сиамских нейронных сетей можно решить общую проблему дисбаланса классов, поскольку сети не требуется слишком много выборок для данного класса в обучающих данных. Более того, новый класс может быть добавлен без обучения всей сети с нуля после того, как сиамская нейронная сеть была обучена и развернута. Модель обучается, изучая, насколько похожи или непохожи пары изображений, образцы из нового класса могут быть добавлены в обученную сиамскую сеть, и обучение может быть возобновлено, поскольку сетевая архитектура

сравнит новые изображения с остальными классами и обновит веса и полностью связанный уровень [8].

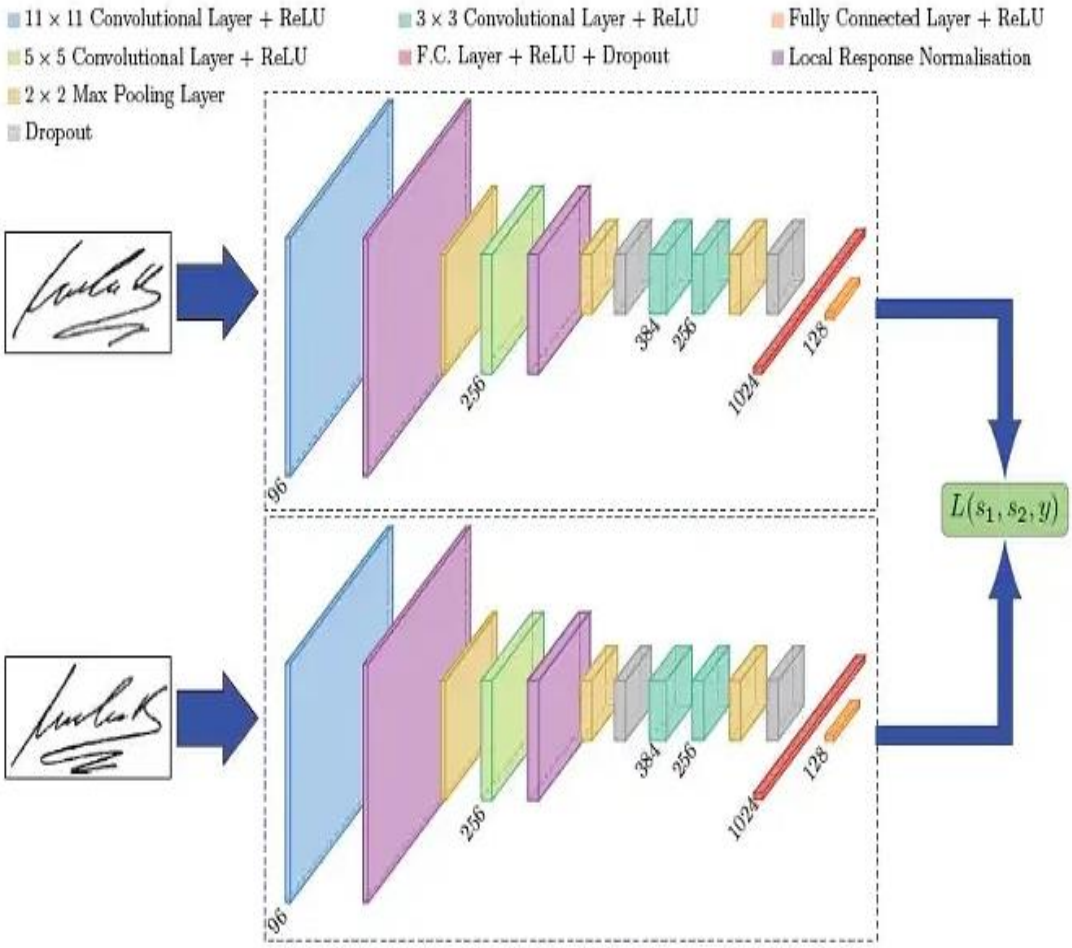

Рисунок 25. – Сиамские сети

Такое поведение уникально для сетевой архитектуры, использующей одноразовое обучение, поскольку для достижения значительной производительности другие категории нейронных сетей пришлось бы обучать с нуля на большом наборе данных, сбалансированном по классам. Но как сиамская сеть обучается на таком небольшом наборе образцов? Давайте посмотрим на архитектуру и на то, как работает процесс обучения в сиамских нейронных сетях. Как описано выше, приведенная ниже архитектура показывает две идентичные подсети, которые составляют сиамскую нейронную сеть. Векторы признаков из обеих сетей сравниваются с использованием функции потерь L. Существует две стратегии обучения сиамской сети с использованием различных функций потерь. Во время обучения векторы признаков должны обладать двумя свойствами, чтобы стратегия обучения с несколькими выстрелами сработала:

Во-первых, векторы признаков похожих и непохожих пар должны быть описательными, информативными и достаточно отличными друг от друга, чтобы можно было эффективно изучать сегрегацию.

Во-вторых, векторы признаков похожих пар изображений должны быть достаточно похожи, а векторы для непохожих пар должны быть достаточно непохожими, чтобы модель могла быстро изучить семантическое сходство.

Чтобы убедиться, что модель может быстро изучать эти векторы признаков, функция потерь должна достаточно сильно стимулировать как изучение сходства, так и различия объектов. Здесь помогает стратегия сиамских нейронных сетей - сравнивая одно изображение со всеми другими изображениями, модель узнает, что такое "похожее" и как определять и распознавать непохожие пары. Для получения такого рода информации кросс-энтропийная потеря не может помочь, поскольку она работает на основе прогнозирования класса. Среднеквадратичные ошибки также не дают достаточной информации, необходимой для нашей цели. Наиболее часто используемые функции потерь - это функция контрастных потерь и функция триплетных потерь. Давайте рассмотрим каждую из них подробно.

Функция контрастных потерь - это функция потерь на основе расстояния, которая обновляет веса таким образом, что два похожих вектора признаков имеют минимальное евклидово расстояние. Для сравнения расстояние между двумя разными векторами максимально. В приведенном ниже уравнении y представляет, являются ли векторы разными, а Dw - евклидово расстояние между векторами. Когда векторы различны (y = 1), функция потерь минимизирует второе слагаемое, для чего Dw должно быть максимальным (поощряйте большее расстояние между разнородными векторами). Мы хотим, чтобы расстояние между этими векторами составляло более по меньшей мере м, и мы избегаем вычислений, если векторы уже разделены m единицами, приняв по умолчанию значение 0. Аналогично, если векторы похожи (y = 0), функция потерь должна минимально имитировать D_w.

$$(1 - Y) * \frac{1}{2} D_w + \frac{1}{2} Y(\max(0, m - D_w))^2 \quad (1)$$

Используя триплетную потерю, мы можем определить, насколько изображение похоже на другие (внутри или за пределами его класса) при сравнении. Сиамская сеть изучает рейтинг сходства, используя оценку, вычисленную таким образом. Для этого потери вычисляются путем сравнения данного изображения (называемого якорным изображением) с положительным изображением (которое похоже на якорное изображение) и отрицательным изображением

(которое отличается от него). Вычисляя внутреннее расстояние для каждой из этих пар, модель знает, как выглядит сходство и насколько данное изображение должно отличаться от других классов. Итак, в приведенном ниже уравнении f (A) - это якорное изображение, а f (P) и f (N) - положительное изображение и отрицательное изображение соответственно. Опять же, чтобы функция потерь минимизировала RHS, член с f (N) должен быть максимальным, а член с f (P) - минимизированным. Это согласуется со стратегией, согласно которой мы хотим, чтобы похожие пары были ближе, а непохожие - дальше друг от друга. α - это просто регулирующий параметр.

$$\mathrm{L(A,P,N)} = \max(\left|\left|\mathrm{f(A)} - \mathrm{f(P)}\right|\right|^2 - \left|\left|\mathrm{f(A)} - \mathrm{f(N)}\right|\right|^2 + \alpha, 0) \quad (2)$$

1.4.8 DeepFace

DeepFace - это алгоритм для распознавания лиц, разработанный командой исследователей Facebook AI Research. Он применяет глубокие сверточные нейронные сети для извлечения высокоуровневых представлений лица и сравнения их с другими лицами. Архитектура DeepFace состоит из нескольких важных компонентов, каждый из которых выполняет специфическую задачу. Первым этапом является слой обнаружения лица, где сверточные слои сканируют входное изображение, чтобы определить и выделить область лица. Это важно, так как на изображении может присутствовать множество объектов, и обнаружение лица позволяет сосредоточиться на соответствующей области. После обнаружения лица следует этап выравнивания лица [9]. Здесь лицо на изображении подвергается преобразованию, чтобы достичь единого положения и масштаба для всех лиц. Это позволяет сделать представления лиц инвариантными к изменениям в положении и ориентации. Следующий важный шаг - слой извлечения признаков. Выравненное лицо проходит через глубокие сверточные слои, которые изучают его особенности на разных уровнях. Это позволяет создать компактное представление лица, содержащее информацию о его уникальных характеристиках, таких как форма лица, расположение глаз, носа, рта и т. д. В конечном счете, извлеченные признаки лица подаются на слой классификации. Сравнивая эти признаки с предварительно сохраненными представлениями лиц в базе данных, алгоритм оценивает сходство между лицами и определяет, к кому принадлежит лицо на входном изображении. DeepFace обучается на огромных наборах данных, содержащих миллионы изображений лиц, чтобы модель смогла обнаруживать широкий спектр лиц и находить общие черты. Этот процесс обучения включает предварительное обучение на больших наборах данных и дообучение на специфических задачах с использованием алгоритмов оптимизации.

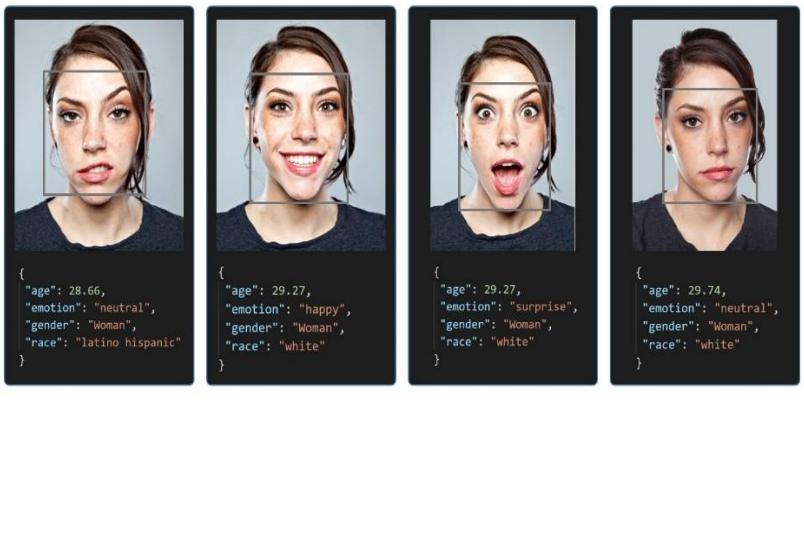

Рисунок 26. – DeepFace

DeepFace имеет преимущества в том, что он работает непосредственно с изображениями лиц и не требует предварительной ручной разметки или выделения ключевых точек. Благодаря использованию глубоких нейронных сетей, DeepFace обладает высокой способностью изучать сложные признаки и обобщать их для классификации и распознавания. Однако важно отметить, что DeepFace и другие алгоритмы распознавания лиц также сталкиваются с рядом вызовов. Некоторые из них включают проблемы с точностью распознавания в условиях с низким качеством изображения, изменчивостью освещения, наличием разных выражений лица и другими факторами, которые могут повлиять на процесс распознавания. Кроме того, возникают этические и приватные вопросы, связанные с использованием данных лиц и защитой личной информации. Тем не менее, DeepFace представляет собой значительный шаг вперед в области распознавания лиц и имеет потенциал для применения в различных сферах, от безопасности и автоматического тегирования фотографий до социальных сетей и биометрических систем.

1.4.9 InsightFace

InsightFace - это передовой алгоритм распознавания лиц, основанный на глубоком обучении и использовании нейронных сетей. Эта технология предлагает точное и эффективное распознавание лиц и имеет широкий спектр применения в различных областях. Основная концепция InsightFace состоит в извлечении уникальных характеристик и представлений лиц,

которые позволяют сравнивать и идентифицировать их. Для этого используются глубокие сверточные нейронные сети, которые обучаются на большом наборе данных изображений лиц [10]. Одним из ключевых компонентов InsightFace является метод ArcFace, который улучшает различимость между лицами. Этот метод использует угловую маржу для более точного разделения представлений лиц в гиперпространстве, что приводит к повышению точности распознавания.

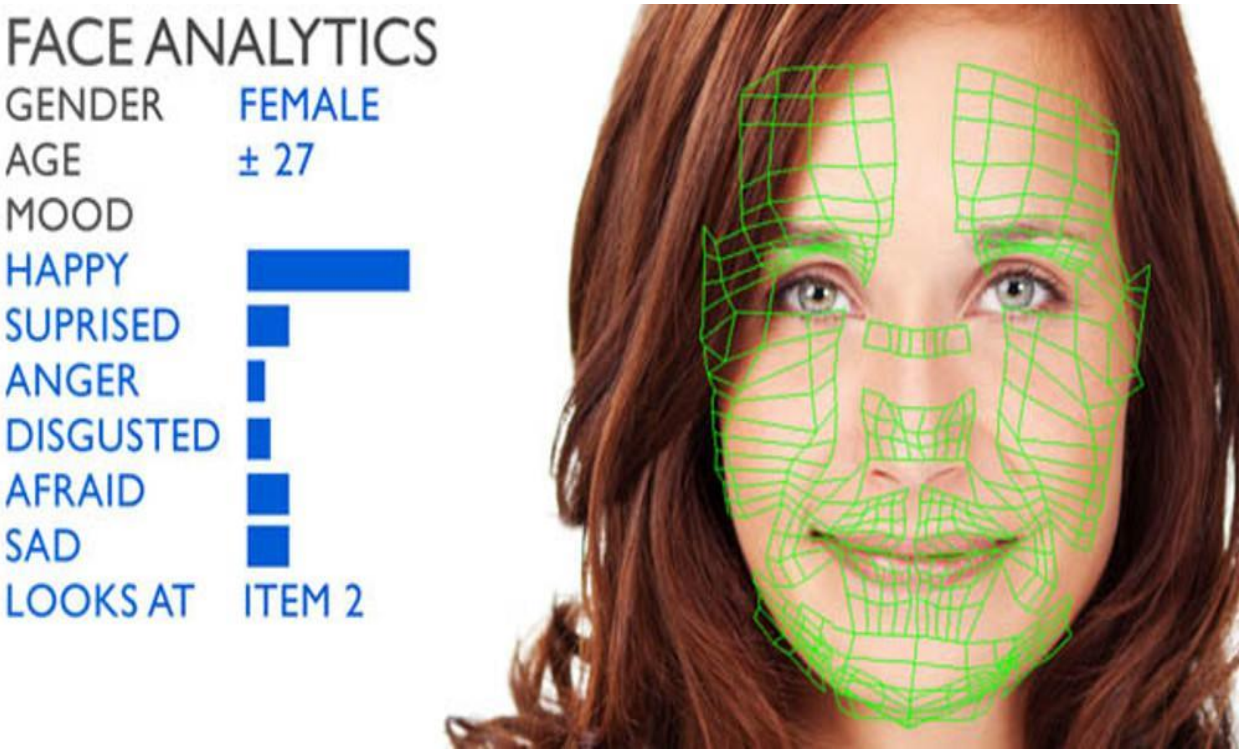

Рисунок 27. – InsightFace

Это особенно полезно при работе с обширными базами данных, содержащими множество лиц. InsightFace также обладает возможностью обнаружения и выравнивания лиц на изображениях. Это помогает получать точные признаки и представления лиц, даже при изменении ракурса или условий освещения. Такой подход обеспечивает более надежное распознавание лиц в различных сценариях. Алгоритм может быть применен в разных областях, включая системы видеонаблюдения, системы безопасности, автоматическую индексацию фотографий, распознавание эмоций и другие. Он обладает значительным потенциалом для улучшения безопасности и эффективности в таких областях, как контроль доступа, идентификация личности и автоматизация процессов. Важно отметить, что точность и эффективность могут зависеть от различных факторов. Качество изображения, условия освещения, присутствие аксессуаров или преград на лице могут влиять на точность

распознавания. Кроме того, использование технологии распознавания лиц вызывает этические и приватные вопросы, и необходимо соблюдать принципы приватности и безопасности данных лиц. В целом, InsightFace является передовым алгоритмом распознавания лиц, который продвигает развитие технологий в области идентификации и анализа лиц, открывая новые возможности в различных приложениях и сферах деятельности.

1.5 Вывод к разделу 1

В первом разделе были рассмотрены известные методы обработки изображения и распознавания лиц на изображении. Также были рассмотрены известные алгоритмы для этих методов. Но эти методы уже всем приелись и их явно можно улучшить, изменив модель сверточной нейронной сети по средством добавления слоев или их изменения. Также точность некоторых из рассмотренных нейронных сетей иногда страдает и главной целью этой работы – это улучшение точности

2 Модель и алгоритмы распознавания лиц

2.1 Модель распознавания лиц

Модель распознания лиц на изображении в исходных кодах программ представлена в виде двух этапов:

1) Этап обучения
2) Этап распознавания.

Каждый этап состоит из двух шагов

1) Обработка входных данных (изображения)
2) Обучение и применение обученной нейронной сети соответственно для каждого этапа

Обучающая выборка состоит из множества изображений с лицами x которые относятся к своим классам y

$$(x_i y_i),\ \ i = 1..n \tag{3}$$

где x_i: исходное изображение;

y_i: класс изображения;

Обучение идет на нейронной сети, которая состоит из двух видов слоев:

1) Сверточные слои
2) Полносвязные слои

В качестве параметров сети используются:

1) Функция активации: ReLu

В отличие от сложных нелинейных функций, которые могут столкнуться с проблемами затухания или взрыва градиентов, простая выпрямленная линейная функция, например ReLU, в некоторых случаях может показывать более преимущественное поведение и достигать лучших результатов. Функция ReLu находится по формуле (4):

$$f(x) = \max(0, x) \tag{4}$$

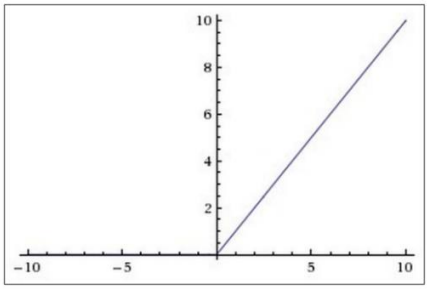

Рисунок 28. – ReLu

Основное свойство ReLu заключается в том, что она является нелинейной функцией, сохраняющей положительные значения и обнуляющей отрицательные значения. Это позволяет модели лучше моделировать сложные нелинейные зависимости между входными данными и выходами. Преимущества ReLu включают простоту вычисления, отсутствие проблемы затухания градиента (gradient vanishing problem), а также способность моделировать разреженность и активацию только некоторых нейронов. Это может приводить к более эффективному обучению модели и улучшению ее обобщающей способности. Однако, у ReLu есть и некоторые недостатки. Во-первых, она не является дифференцируемой в нуле, что может затруднять использование градиентных методов оптимизации, которые требуют вычисления градиента. В таких случаях обычно используются аппроксимации или вариации ReLu, такие как Leaky ReLu или Parametric ReLu. Во-вторых, ReLu может быть неактивной для отрицательных значений и, следовательно, не способна передавать градиент во время обратного распространения ошибки, что может привести к нейронам, не обновляющимся в процессе обучения («мертвые нейроны»). Не смотря на некоторые ограничения, функция активации ReLu остается популярным выбором для многих типов нейронных сетей, включая сверточные нейронные сети (CNN) и глубокие нейронные сети (DNN).

2) Функция активации на последнем слое: Softmax

Softmax (софтмакс) - это функция активации, которая обычно используется в конце нейронных сетей для решения задачи многоклассовой классификации. Она преобразует произвольные вещественные числа в вероятностные значения, сумма которых равна 1. Процесс работы softmax можно представить следующим образом: для каждого элемента вектора x вычисляется экспонента, затем каждое значение делится на сумму экспонент всех элементов. Это позволяет получить значения, интерпретируемые как вероятности каждого класса.

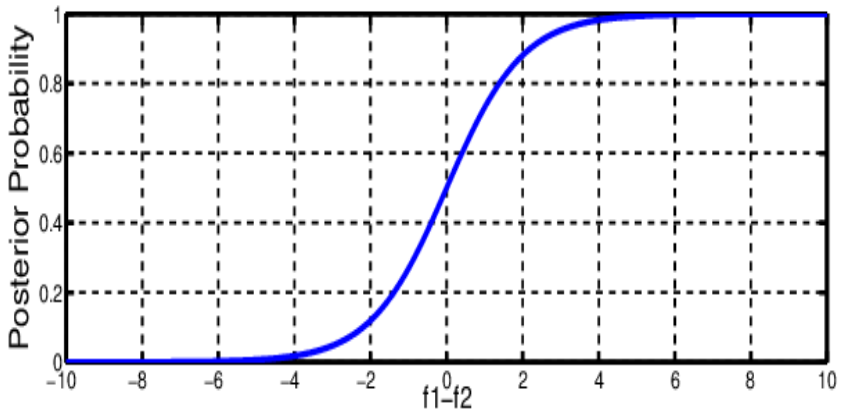

Рисунок 29. – Softmax

Преимуществом использования softmax является то, что она генерирует вероятностное распределение по классам, что удобно для многоклассовой классификации. Например, если у вас есть 3 класса (A, B, C), softmax преобразует выходы нейронной сети в вероятности (например, A: 0.2, B: 0.3, C: 0.5), которые можно интерпретировать как "вероятность принадлежности" каждого класса. Эта функция также используется в связке с функцией потерь, такой как категориальная кросс-энтропия, для обучения нейронных сетей в задачах классификации. Она позволяет вычислять градиенты и обновлять веса сети, чтобы минимизировать ошибку классификации и улучшить обобщающую способность модели.

3) Функция оптимизации: SGD

SGD является вариантом градиентного спуска, который позволяет эффективно обучать модели машинного обучения на больших наборах данных. Вместо того, чтобы использовать весь набор данных для вычисления градиента и обновления параметров модели, SGD выбирает случайный обучающий пример или небольшой пакет примеров (также называемый мини-пакетом) на каждой итерации. Этот случайный выбор примеров вносит стохастичность (случайность) в процесс оптимизации. В результате SGD способен быстрее справляться с вычислительными требованиями больших наборов данных, поскольку требуется обработка только одного примера или небольшого пакета примеров на каждой итерации. Алгоритм градиентного спуска обновляет значения весов модели, основываясь на градиенте (направлении наискорейшего убывания) функции потерь в данной точке. Параметр скорости обучения (learning rate) определяет размер шага, с которым обновляются веса. Он контролирует величину изменений весов на каждом шаге и может влиять на скорость сходимости и качество оптимизации.

SGD является популярным алгоритмом оптимизации в машинном обучении благодаря своей вычислительной эффективности. Однако он также может иметь некоторые недостатки, такие как шумность обновлений весов из-за случайности выбора примеров и возможность застревания в локальных минимумах функции потерь. Для борьбы с этими проблемами существуют различные вариации SGD, такие как момент (momentum), адаптивный скорость обучения (adaptive learning rate) и другие. Однако SGD остается основным инструментом для оптимизации моделей машинного обучения, и его использование позволяет эффективно обучать модели на больших наборах данных.

$$\omega_{t+1} = \omega_t - \alpha * \frac{\delta L}{\delta \omega} \quad (5)$$

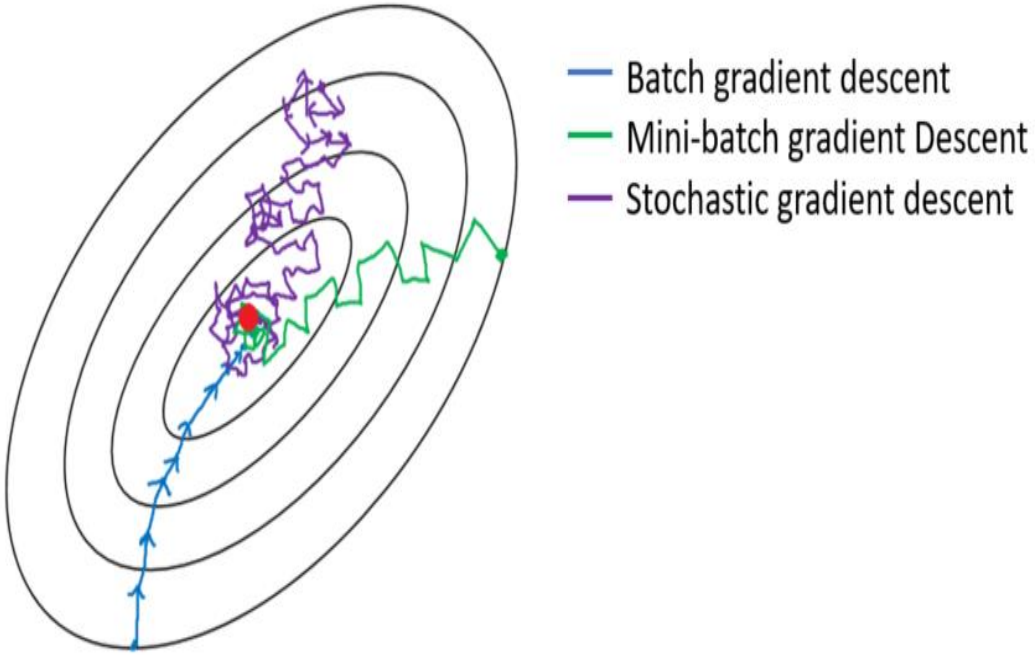

Рисунок 30. – SGD

4) Функция потерь: Категориальная кросс-энтропия

Категориальная кросс-энтропия (Categorical Cross-Entropy) является функцией потерь, используемой в задачах классификации, где требуется предсказывать вероятности для нескольких классов. Она широко применяется в машинном обучении для оценки различия между предсказанными вероятностями классов и истинными метками.

$$\text{CrossEntropy}(\text{p}_{true}, \text{p}_{pred} \sum_{i=1}^{N} \text{p}_{true}[i] * \ln(\text{p}_{pred}[i]) \quad (6)$$

Интерпретация функции потерь заключается в том, что мы штрафуем модель за большое расхождение между предсказанными и истинными вероятностями классов. Чем больше различие между этими вероятностями, тем выше значение кросс-энтропии и тем больший штраф получает модель. Целью оптимизации является минимизация значения категориальной кросс-энтропии, чтобы модель предсказывала вероятности классов, максимально приближенные к истинным меткам. Для этого используются методы градиентного спуска или его вариаций, которые позволяют обновлять веса модели, чтобы минимизировать функцию потерь. Категориальная кросс-энтропия является часто используемой функцией потерь в задачах многоклассовой классификации, где требуется предсказывать вероятности для нескольких классов, таких как распознавание изображений или классификация текстовых данных

Этап обучения:

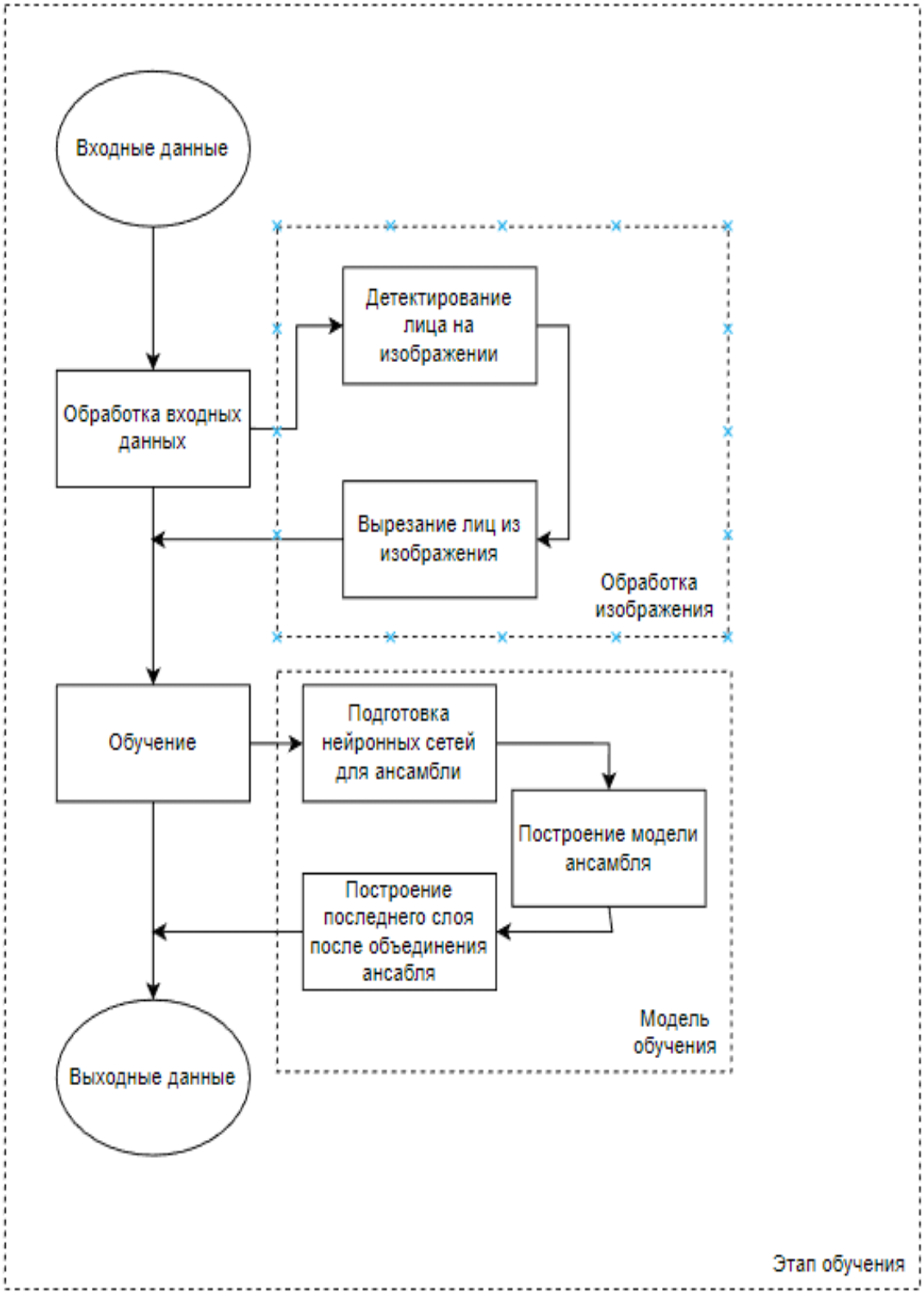

Рисунок 31. – Этап Обучения

Этап распознавания:

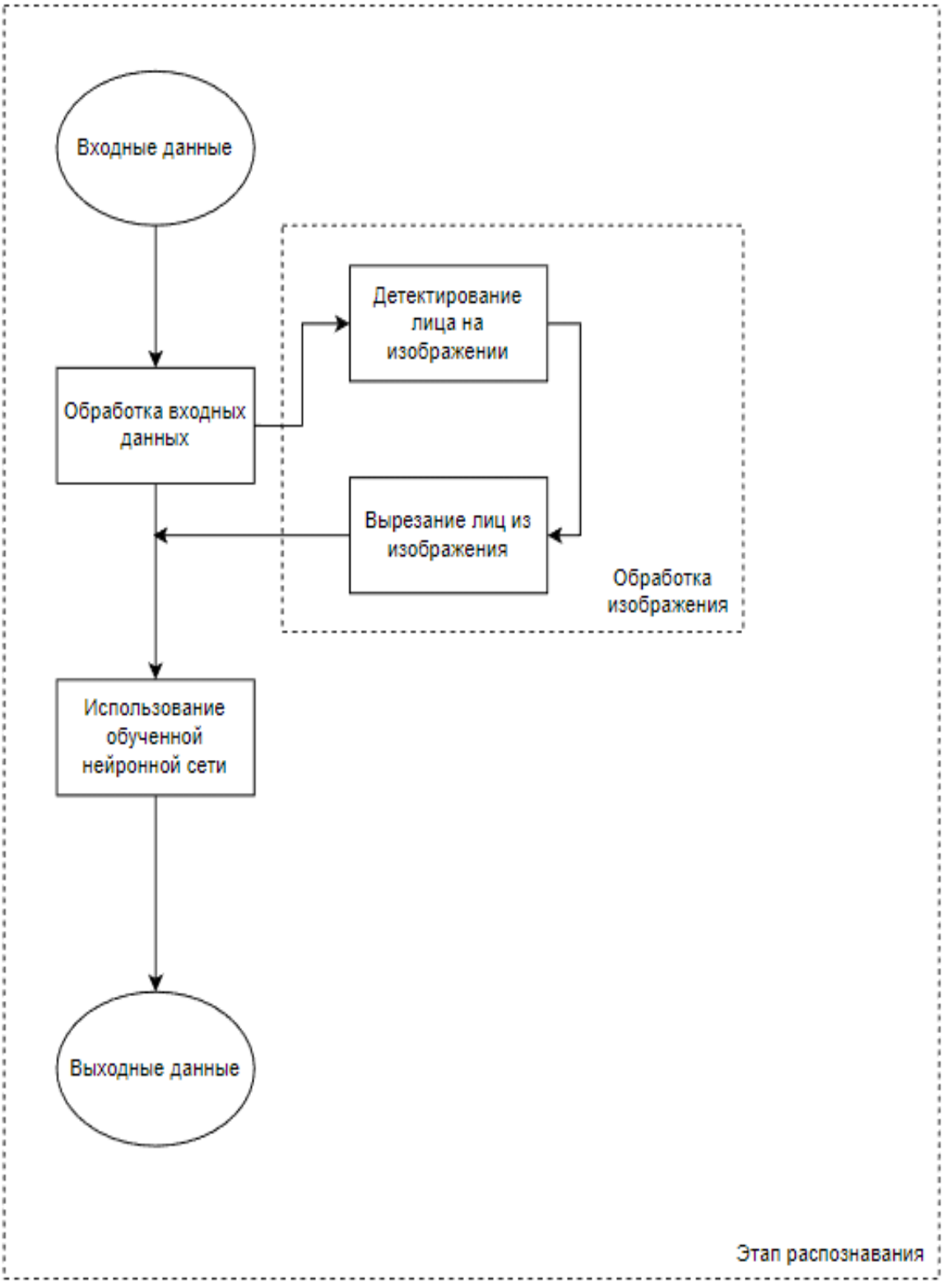

Рисунок 32. – Этап распознавания

Общий алгоритм для распознания лица выглядит следующем образом:

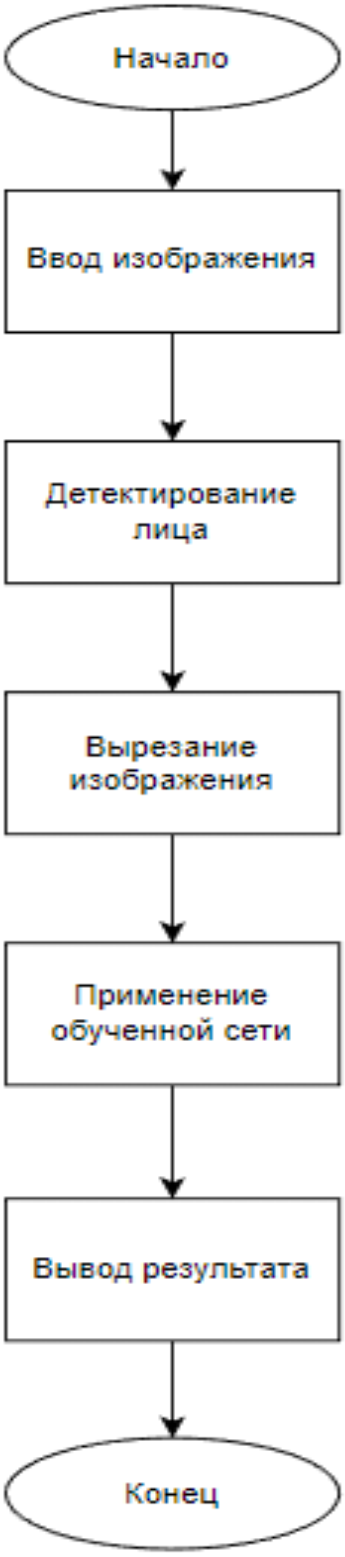

Рисунок 33. – Блок-схема распознавания лиц

2.2 Алгоритм предварительной обработки данных

Для детектирования лиц на изображении был выбран каскадный алгоритм Хаара. Библиотека OpenCV предлагает инструмент, использующий каскады.

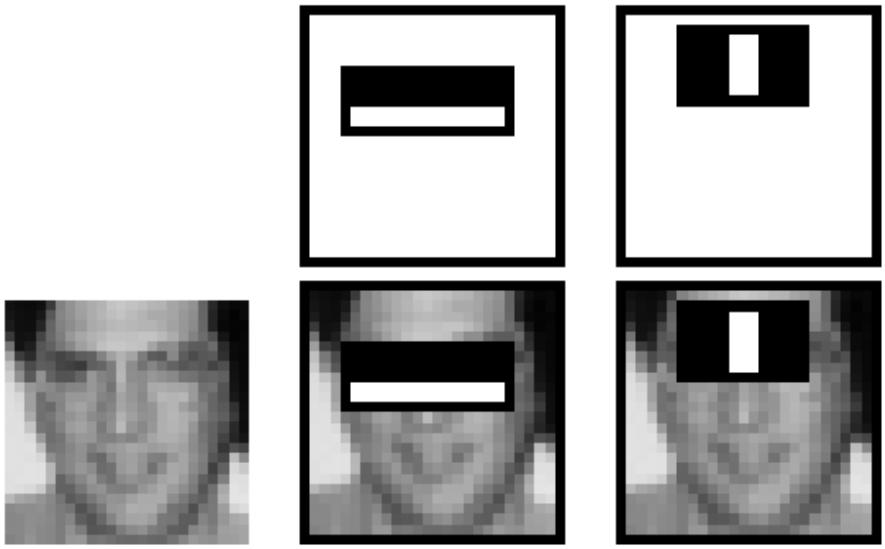

Рисунок 34. – Алгоритм Хаара

Первоначальная версия алгоритма Виолы-Джонса использовала только простые примитивы без учета поворотов. Для вычисления значений признаков была использована разность сумм яркостей пикселей между двумя подобластями. В дальнейшем метод был развит и в него были внесены изменения, включая примитивы с наклоном на 45 градусов и несимметричные конфигурации. Кроме того, вместо обычной разности, каждой подобласти был присвоен вес, и значения признаков вычислялись как взвешенная сумма пикселей из разных типов подобластей.

Для детектирования лиц на изображении был выбран каскадный алгоритм Хаары. Библиотека OpenCV предлагает инструмент, использующий каскады.

$$\text{feature} = \sum_{i=1}^{N} \omega_i * \text{RectSum}(\text{r}_i) \quad (7)$$

2.3 Алгоритмы распознания лиц

В моей работе были использованы такие известные нейронные сети, как ResNet, MobileNet, VGG-16.

1) ResNet – остаточная сеть

ResNet (Residual Neural Network) - это тип архитектуры нейронных сетей, разработанный для решения проблемы затухания градиента (vanishing gradient problem) при обучении глубоких нейронных сетей. Он был представлен в статье "Deep Residual Learning for

Image Recognition" в 2015 году. Основная идея ResNet заключается в использовании блоков с пропусками (skip connections), которые позволяют пропустить несколько слоев и прямо передавать информацию от одного слоя к следующему. Это позволяет градиенту проходить через сеть более эффективно, снижая вероятность затухания градиента и улучшая обучение глубоких моделей.

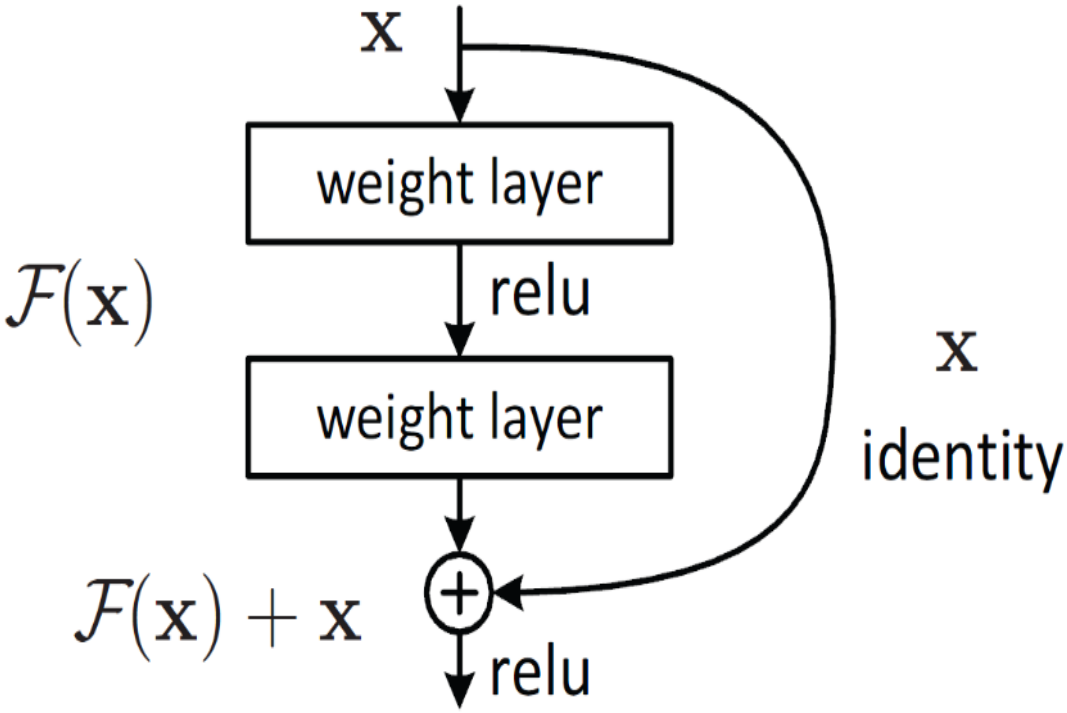

Рисунок 35. – Особенность ResNet

Shortcut Connection (пропускающее соединение) представляет собой прямое соединение от входа блока к выходу блока. Это делает возможным пропускать слои и передавать информацию без изменений напрямую. Если размеры входа и выхода блока отличаются, то Shortcut Connection может выполнять операции свертки или пулинга для приведения размерности входа к размерности выхода. ResNet имеет различные варианты архитектур, включая ResNet-18, ResNet-34, ResNet-50, ResNet-101 и ResNet-152. Число в названии указывает общее количество слоев в сети. Более глубокие модели, такие как ResNet-101 и ResNet-152, имеют большее число слоев и обычно показывают лучшую производительность на сложных задачах, но требуют больше вычислительных ресурсов для обучения и инференса. ResNet и его варианты стали популярными в области компьютерного зрения и достигли впечатляющих результатов на различных задачах, включая классификацию изображений, обнаружение объектов и сегментацию изображений.

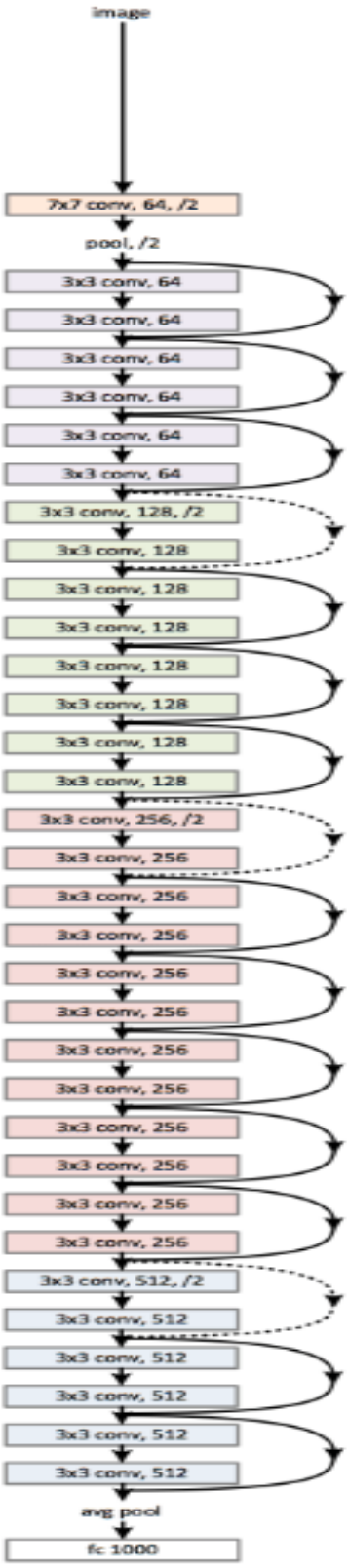

Рисунок 36. – ResNet

2) MobileNet

MobileNet - это архитектура нейронной сети, разработанная для эффективного выполнения задач компьютерного зрения на мобильных устройствах с ограниченными ресурсами, такими как смартфоны и встроенные системы. Она была представлена в статье "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" в 2017 году.

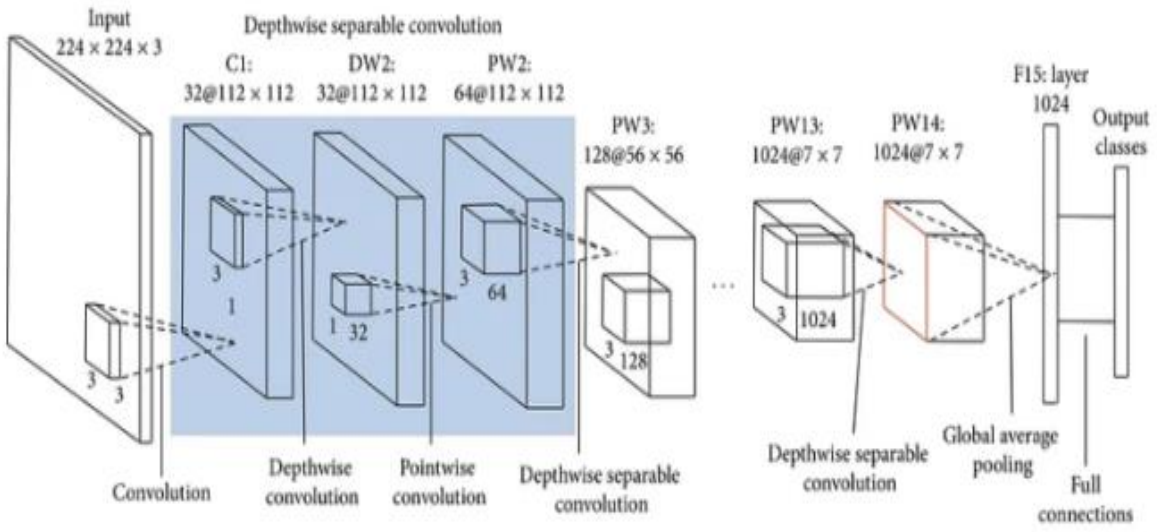

Рисунок 37. – MobileNet

Главная цель MobileNet - достичь высокой производительности при малом количестве вычислительных операций и параметров модели. Для этого в архитектуре MobileNet используются две основные техники: глубокая свертка (depthwise separable convolution) и использование сверточных слоев с малым количеством фильтров. Глубокая свертка состоит из двух шагов: сначала производится свертка по каждому каналу входных данных отдельно с использованием фильтров размера 1x1 (глубокая свертка по каналам), затем применяется свертка с фильтрами размера nxn (глубокая свертка по пространственным размерам). Таким образом, глубокая свертка позволяет снизить количество вычислений, по сравнению с обычной сверткой. Сверточные слои с малым количеством фильтров (обычно 1x1) используются для уменьшения числа параметров и операций в модели. Они выполняют сжатие и изменение размерности признакового пространства, что помогает сократить вычислительные затраты. Архитектура MobileNet имеет различные варианты, такие как MobileNetV1, MobileNetV2 и MobileNetV3, каждый из которых внес свои улучшения и оптимизации. В более поздних версиях MobileNet были добавлены дополнительные слои и блоки, чтобы улучшить точность и эффективность модели. MobileNet успешно применяется в различных задачах компьютерного зрения, включая

классификацию изображений, обнаружение объектов, сегментацию изображений и многие другие. Благодаря своей эффективности и низким требованиям к вычислительным ресурсам, MobileNet стал популярным выбором для развертывания моделей машинного обучения на мобильных платформах.

3) VGG-16

VGG-16 (Visual Geometry Group-16) - это архитектура глубокой нейронной сети, представленная в 2014 году исследовательской группой Visual Geometry Group (VGG) при Университете Оксфорд. VGG-16 является одной из популярных моделей для задач компьютерного зрения, таких как классификация изображений. Архитектура VGG-16 состоит из 16 слоев, включая 13 сверточных слоев и 3 полносвязных слоя. Все сверточные слои имеют фильтры размером 3x3, а max pooling применяется после каждых двух сверточных слоев для уменьшения размерности. Активация в сети осуществляется с использованием функции активации ReLU (Rectified Linear Unit).

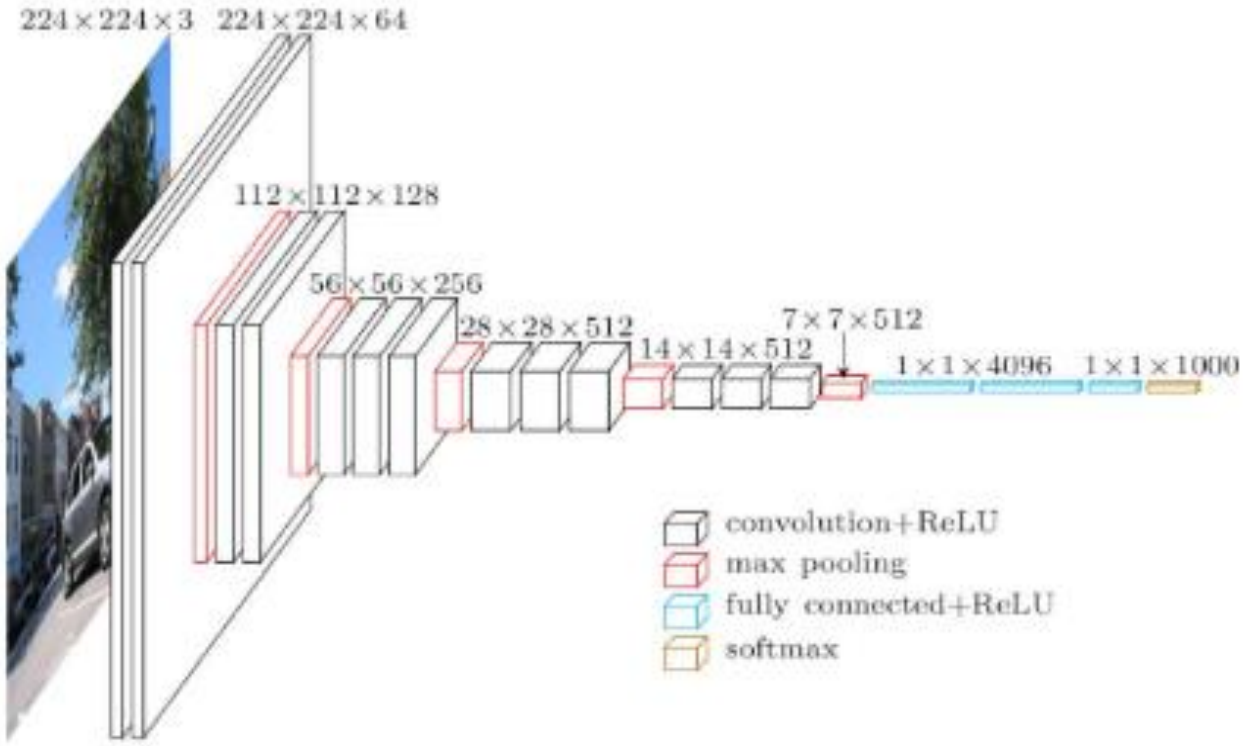

Рисунок 38. – VGG-16

Основная особенность VGG-16 - глубокая архитектура с повторяющимися сверточными блоками. Вместо использования больших сверточных фильтров, VGG-16 использует малые фильтры размером 3x3. Это позволяет модели изучать более мелкие детали изображений и получать более точные признаки. После сверточных слоев в VGG-16 следуют

полносвязные слои, которые преобразуют выходные данные сверточных слоев в конечные предсказания классов. Последний полносвязный слой использует функцию активации Softmax для получения вероятностей принадлежности изображения к каждому классу. VGG-16 имеет высокую точность при классификации изображений, но за счет своей глубокой структуры требует большого количества параметров и вычислительных ресурсов для обучения и работы. Она широко использовалась в исследованиях и соревнованиях по компьютерному зрению, и ее архитектура стала основой для других моделей, таких как VGG-19 и VGG-Net с другими вариантами количества слоев. Обучение VGG-16 может быть затратным в плане вычислительных ресурсов, но предобученные веса VGG-16 на больших наборах данных, таких как ImageNet, доступны для использования и могут быть перенесены для решения задач классификации и других задач компьютерного зрения.

Также в работе используется механизм ансамблирования нейронной сети. Ансамбль нейронных сетей (Neural Network Ensemble) - это метод комбинирования нескольких нейронных сетей для решения задач машинного обучения. Вместо использования отдельной нейронной сети, ансамбль объединяет несколько независимых нейронных сетей, называемых базовыми моделями или участниками ансамбля.

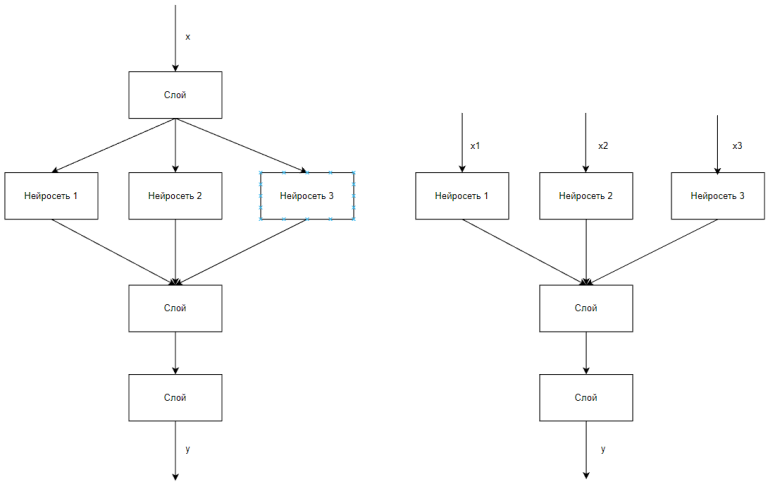

Рисунок 39. – Ансамбль нейронных сетей

Каждая нейронная сеть в ансамбле обучается на одной и той же задаче с использованием одного и того же набора данных или различных подмножеств набора данных. Каждая модель может иметь различную архитектуру или гиперпараметры, что позволяет ансамблю охватить различные аспекты данных и повысить обобщающую способность. В процессе использования ансамбля нейронных сетей для классификации или регрессии, каждая

модель делает прогноз, а итоговый прогноз получается путем комбинирования прогнозов всех моделей [6]. Существует несколько способов комбинирования прогнозов, таких как голосование большинством, усреднение или взвешенное голосование. Преимущества ансамбля нейронных сетей включают повышение точности прогнозирования, улучшение устойчивости модели к различным вариациям данных, улучшение обобщающей способности модели и снижение риска переобучения. Однако использование ансамбля требует большего объема вычислительных ресурсов и времени для обучения и предсказания. В некоторых случаях использование разных нейронных сетей может улучшить результат обучения за счет возможного покрытия неточностей одной нейронный сети другой нейронной сетью. Также в ансамбле нейронных сетей имеется возможность обучать разные нейронные сети с разными датасетами в одной системе.

2.4 Программная реализация

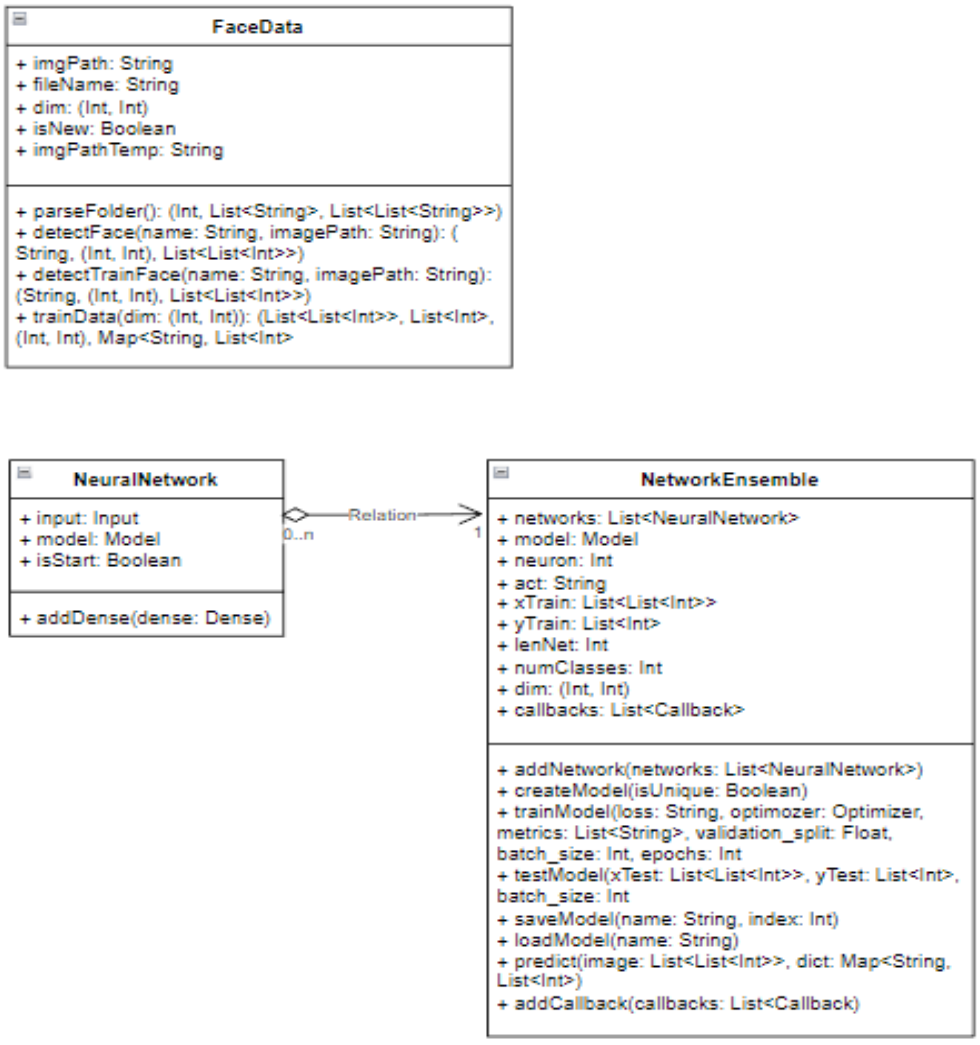

Рисунок 40. – Диаграмма классов программы

Для формирования ансамбля нейронных сетей для распознавания лиц была написана программа, состоящая из трех частях:

1) Подготовка данных для сверточной нейронной сети

Эта часть программы обрезает изображение лица из всего изображения и сохраняет его для дальнейшего использования. За это отвечает класс FaceData:

1. Входные параметры класса:

a. imagePath – имя директории, в которой находятся изображения (по умолчанию равна значению пустой строки).

b. filename – имя файла, в который будет записываться информация ою изображениях в виде (класс, путь) (по умолчанию равна значению пустой строки).

c. isNew – флаг, показывающий использование данных в экземпляре (по умолчанию равен истинному значению)

2. Дополнительные аргументы:

a. dim – кортеж, показывающий размер изображения, который должен быть у изображения лица. (по умолчанию равен значению (120, 120) и изменяется только при вызове метода trainData с параметром)

b. imgPathTemp – имя директории, куда будут складываться изображения лица.

3. Методы:

a. parseFolder() – метод, который считывает все изображения из папки. Входные параметры отсутствуют. Результатом работы функции являются: количество классов, имя классов, массив путей до изображения

```
def parseFolder(self):
    folder = self.imgPath if self.isNew else self.imgPathTemp
    folderPaths = os.listdir(folder)
    photoWithName = []
    print("Read Images: ")
    for i, path in enumerate(tqdm(folderPaths)):
        imagePaths = os.listdir(folder + '\\' + path)
        for j, imagePath in enumerate(imagePaths):
            filePath = folder + '\\' + path + '\\' + imagePath
            if os.path.isfile(filePath):
                photoWithName.append([path, filePath])
    print("End Read.\n")
    return len(folderPaths), folderPaths, photoWithName
```

Рисунок 41. – Функция parseFolder

b. detectTrainFace(name, imagePath) – метод, который обрезает изображения лица из исходного изображения и сохраняет его в другую папку, если он в ней отсутствует. Входные параметры: name (имя класса) и imagePath (путь к изображению). Результатом работы функции являются: имя класса, размер изображения лица, изображения лица

```
def detectTrainFace(self, name, imagePath):
    images = []
    image = cv2.imread(imagePath)
    imgFolder = self.imgPathTemp + '\\' + name
    fileName = os.path.basename(imagePath)
    imagePathNew = imgFolder + '\\' + fileName

    if self.isNew:
        grayImage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        haarCascade = cv2.CascadeClassifier('Haarcascade_frontalface_default.xml')
        facesRect = haarCascade.detectMultiScale(grayImage, 1.1, 9)

        for (x, y, w, h) in facesRect:
            face = image[y:y+h, x:x+w]
            faceDim = cv2.resize(face, self.dim, interpolation = cv2.INTER_AREA)
            images.append(faceDim)

        if not os.path.exists(self.imgPathTemp):
            os.makedirs(self.imgPathTemp)
        if not os.path.exists(imgFolder):
            os.makedirs(imgFolder)
        if not os.path.exists(imagePathNew):
            try:
                cv2.imwrite(imagePathNew, images[0])
            except:
                print("Failed image {}, {}".format(name, imagePath))
    else:
        images = [cv2.resize(image, self.dim, interpolation = cv2.INTER_AREA)]

    return name, self.dim, images
```

Рисунок 42. – Функция detectTrainFace

c. detectFace(name, imagePath) - метод, который обрезает изображения лица из исходного изображения. Входные параметры: name (имя класса) и imagePath (путь к изображению). Результатом работы функции являются: имя класса, размер изображения лица, изображения лица.

```
def detectFace(self, name, imagePath):
    images = []
    image = cv2.imread(imagePath)
    grayImage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    haarCascade = cv2.CascadeClassifier('Haarcascade_frontalface_default.xml')
    facesRect = haarCascade.detectMultiScale(grayImage, 1.1, 9)

    for (x, y, w, h) in facesRect:
        face = image[y:y+h, x:x+w]
        faceDim = cv2.resize(face, self.dim, interpolation = cv2.INTER_AREA)
        images.append(faceDim)

    return name, self.dim, images
```

Рисунок 43. – Функция detectFace

d. trainData(dim) – метод, который из изображений делает входной и выходной датасет. Входные параметры: dim (размер изображения лица). Результатом программы являются: входные и выходные параметры для нейронной сети, размер изображения лица, словарь с расшифровкой выходных данных.

```
def trainData(self, dim = (120, 120)):
    self.dim = dim
    personNum, persons, data = self.parseFolder()

    print("Create Image Data: ")
    x = []
    y = []
    yDict = {}
    for i, image in enumerate(tqdm(data)):
        name, dim, images = self.detectTrainFace(image[0], image[1])

        personIndex = persons.index(name)
        try:
            x.append(images[0] / 255.0)
            y.append(personIndex)
            yDict[name] = personIndex
        except:
            print("Failed Image Data {}, {}".format(image[0], image[1]))

    y = to_categorical(np.array(y), personNum)
    personCat = to_categorical([i for i in range(personNum)], personNum)

    for i, name in enumerate(persons):
        yDict[name] = personCat[i]
    self.isNew = False
    print("End Image Data\n ")
    return x, y, dim, yDict
```

Рисунок 44. – Функция trainData

2) Составление моделей

Эта часть программы служит для составления моделей для ансамбля до его объединения. Этим управляет класс NeuralNetwork

1. Входные параметры:

a. input – формат входных данных нейронной сети

2. Методы:

a. addDense(dense) – метод, который добавляет слой к модели. Входные параметры: слой. Результатом является добавления слоя в модель нейронной сети

```
def addDense(self, dense):
    if self.isStart:
        self.model = dense(self.input)
        self.isStart = False
    else:
        self.model = dense(self.model)
```

Рисунок 45. – Функция addDense

3) Действия с моделями

Эта часть программы служит для выполнения разных действий с моделями нейронной сети. Этим управляет класс NetworkEnsemble

1. Входные параметры:

a. neuron – стандартное значения количества нейронов для слоя, который будет идти после объединения ансамбля

b. act – функция активации для слоя, который будет идти после объединения ансамбля

c. xTrain – входные данные для обучения

d. yTrain – выходные данные для обучения

e. numClasses – количество классов

f. dim – размер изображений лиц

2. Методы

a. addNetwork(networks) – метод, добавляющий нейронную сеть в ансамблю. Входные параметры: networks (модели или массив моделей) Результатом является добавления нейронной сети в ансамбль.

```
def addNetwork(self, networks):
    if type(networks)==list:
        for net in networks:
            self.networks.append(net)
    else:
        self.networks.append(networks)

    self.lenNet = len(self.networks)
```

Рисунок 46. – Функция addNetwork

b. addCallBack(callbacks) – метод, оставляющий отчеты после каждой эпохи сети. Входные параметры: callbacks (отчет или отчеты). Результатом является добавления отчеты к нейронной сети.

```
def addCallback(self, callbacks):
    if type(callbacks)==list:
        for callback in callbacks:
            self.callbacks.append(callback)
    else:
        self.callbacks.append(callback)
```

Рисунок 47. – Функция addCallback

c. createModel(isUnique) – метод, объединяющий нейронные сети и добавляющий полно связные слои. Входные параметры: isUnique – флаг, отличающий готовую нейросеть от модели или нейросети для ансабля. Результатом является готовая модель для обучения ансамбля

```
def createModel(self, isUnique = False):
    if (len(self.networks) > 1):
        out = Concatenate()([net.model for net in self.networks])
        out = Dense(self.neuron * 2, activation = self.act)(out)
        out = Dropout(0.25)(out)

        out = Dense(self.neuron // 2, activation = self.act)(out)
        out = Dropout(0.25)(out)

        out = Dense(self.neuron // 4, activation = self.act)(out)
        out = Dense(self.numClasses, activation = 'softmax')(out)

        self.model = Model([net.input for net in self.networks], out)

    else:
        out = self.networks[0].model
        if isUnique:
            self.model = Model(self.networks[0].input, out)
        else:
            out = Dense(self.neuron * 2, activation = self.act)(out)
            out = Dropout(0.25)(out)

            out = Dense(self.neuron // 2, activation = self.act)(out)
            out = Dropout(0.25)(out)

            out = Dense(self.neuron // 4, activation = self.act)(out)
            out = Dense(self.numClasses, activation = 'softmax')(out)

            self.model = Model([net.input for net in self.networks], out)
```

Рисунок 48. – createModel

d. trainModel(loss, optimizer, metrics, validation_split, batch_size, epochs) – метод, обучающий готовую модель. Входные параметры: функция потеря, функция активации, метрики, доля входных данных для проверочной выборке, пакет обучения, количество эпох. Результатом является обученная модель.

```
def trainModel(self, loss, optimizer, metrics, validation_split, batch_size, epochs):
    self.model.summary()

    self.model.compile(loss=loss,
                       optimizer=optimizer,
                       metrics=metrics)

    print("\nTrain Model:")
    self.model.fit([np.array(self.xTrain) for net in self.networks],
                   np.array(self.yTrain),
                   validation_split = validation_split,
                   batch_size = batch_size,
                   epochs = epochs,
                   callbacks = self.callbacks)
```

Рисунок 49. – Функция trainModel

e. testModel(xTest, yTest, batch_size) - метод, тестирующий готовую модель. Входные параметры: входные тестовые данные, выходные тестовые данные, пакет обучения. Результатом является тестируемая модель.

```
def testModel(self, xTest, yTest, batch_size):
    print("\nTest Model:")
    self.model.evaluate([np.array(xTest) for net in self.networks],
                        np.array(yTest),
                        batch_size=batch_size)
```

Рисунок 50. – Функция testModel

f. saveModel(name, index) – метод сохранения архитектуры и весов нейронной сети. Входные параметры: имя, индекс для обхода дубликатов. Результатом является сохранение архитектуры и весов нейронной сети

```
def saveModel(self, name, index = 0):
    fname = '{}_{}'.format(name, index)
    fileName = "{}.h5".format(fname)
    print(fileName)
    netFileName = "{}.txt".format(fname)
    try:
        if os.path.isfile(fileName):
            self.saveModel(name, index + 1)
        else:
            self.model.save(fileName)
            f = open(netFileName, 'w')
            f.write("{}".format(self.lenNet))
            f.close()
    except:
        pass
```

Рисунок 51. – Функция saveModel

g. loadModel(name) – метод, загружающий модель. Входные параметры: имя файла. Результатом является изменение модели на загружаемую с файла

```
def loadModel(self, name):
    f = open("{}.txt".format(name), 'r')
    self.model = load_model("{}.h5".format(name))
    for line in f:
        self.lenNet = int(line[:-1])
    f.close()
```

Рисунок 52. – Функция loadModel

h. predict(image) – метод, выдающий результат предсказания нейронной сети. Входные параметры: изображение. Результатом является класс.

```
def predict(self, image):
    (w, h) = self.dim
    if (self.lenNet == 1):
        predict = self.model.predict(np.array(image))
    else:
        predict = self.model.predict([np.array(image) for i in range(self.lenNet)])
    print(predict)
    return(predict)
```

Рисунок 53. – Функция predict

2.5 Набор данных

В работе был использован датасет состоящий из 1500 изображений лиц, которые поделены на 20 классов. На обучающую выборку пришлось 1250 изображений, а на тестовую 250 изображений. В каждом изображении было вырезано лицо и приведено к размеру 120x120. Каждый класс изображений был помещен в отдельную папку.

Рисунок 54. – Датасет

2.6 Вывод к разделу 2

В данном разделе была рассмотрена модель нейронной сети для распознания лиц. Также были подробно рассмотрены каждые шаги, начиная с детектирования лица на изображении, заканчивая рассмотрение алгоритмов обучения. Были рассмотрены такие популярные алгоритмы распознавания, как VGG-16, ResNet, MobileNet. У каждого из этих алгоритмов свой подход, свои преимущества и недостатки и при их совмещении недостатки каждого компенсируются и сеть из их ансамбля будет давать более точные распознавания. Также был рассмотрен механизм ансамблирования, который позволяет в одной архитектуре совмещать несколько моделей, которые будут обучаться параллельно друг от друга. Был рассмотрен датасет, на котором будут обучаться нейронные сети, который состоит из 1500 изображений, поделенный на 20 классов.

3 Вычислительный эксперимент

3.1 Постановка задачи вычислительного эксперимента

В ходе работы надо удостоверится, что ансамбли нейронных сетей могут улучшить результат обучения нейронной сети.

3.2 Эксперимент 1

В первом эксперименте была использована нейронная сеть с batch size = 8 и количеством эпох равным 10.

1) ResNet

```
Train Model:
Epoch 1/10
6/6 [==============================] - 43s 6s/step - loss: 0.2250 - accuracy: 0.4785
Epoch 2/10
6/6 [==============================] - 35s 6s/step - loss: 0.2168 - accuracy: 0.4570
Epoch 3/10
6/6 [==============================] - 35s 6s/step - loss: 0.1993 - accuracy: 0.5000
Epoch 4/10
6/6 [==============================] - 36s 6s/step - loss: 0.1934 - accuracy: 0.5269
Epoch 5/10
6/6 [==============================] - 35s 6s/step - loss: 0.2107 - accuracy: 0.4839
Epoch 6/10
6/6 [==============================] - 35s 6s/step - loss: 0.1904 - accuracy: 0.5269
Epoch 7/10
6/6 [==============================] - 35s 6s/step - loss: 0.1836 - accuracy: 0.5645
Epoch 8/10
6/6 [==============================] - 35s 6s/step - loss: 0.1743 - accuracy: 0.5645
Epoch 9/10
6/6 [==============================] - 35s 6s/step - loss: 0.1818 - accuracy: 0.5538
Epoch 10/10
6/6 [==============================] - 35s 6s/step - loss: 0.1762 - accuracy: 0.5699
netResNet_0.h5

Test Model:
3/3 [==============================] - 3s 463ms/step - loss: 0.2239 - accuracy: 0.4615
```

Рисунок 55. – ResNet (Эксперимент 1)

Как видим обучение прошло до 56%, а на тестовых данных был показан результат в 46%.

2) VGG-16

```
Train Model:
Epoch 1/10
6/6 [==============================] - 59s 10s/step - loss: 0.2222 - accuracy: 0.4086
Epoch 2/10
6/6 [==============================] - 58s 10s/step - loss: 0.2221 - accuracy: 0.3602
Epoch 3/10
6/6 [==============================] - 58s 10s/step - loss: 0.2221 - accuracy: 0.3710
Epoch 4/10
6/6 [==============================] - 59s 10s/step - loss: 0.2221 - accuracy: 0.3656
Epoch 5/10
6/6 [==============================] - 58s 10s/step - loss: 0.2220 - accuracy: 0.3548
Epoch 6/10
6/6 [==============================] - 59s 10s/step - loss: 0.2220 - accuracy: 0.3548
Epoch 7/10
6/6 [==============================] - 59s 10s/step - loss: 0.2220 - accuracy: 0.3548
Epoch 8/10
6/6 [==============================] - 59s 10s/step - loss: 0.2220 - accuracy: 0.3710
Epoch 9/10
6/6 [==============================] - 58s 10s/step - loss: 0.2219 - accuracy: 0.3548
Epoch 10/10
6/6 [==============================] - 60s 10s/step - loss: 0.2219 - accuracy: 0.3602
netVGG_0.h5

Test Model:
3/3 [==============================] - 3s 840ms/step - loss: 0.2216 - accuracy: 0.4615
```

Рисунок 56. – VGG-16 (Эксперимент 1)

Как видим обучение прошло до 36%, а на тестовых данных был показан результат в 46%.

3) MobileNet

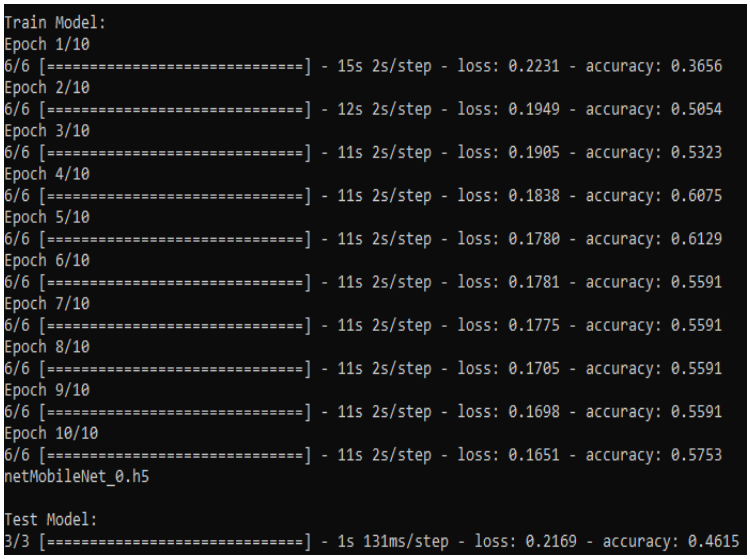

Рисунок 57. – MobileNet (Эксперимент 1)

Как видим обучение прошло до 57%, а на тестовых данных был показан результат в 46%.

4) ResNet + VGG-16

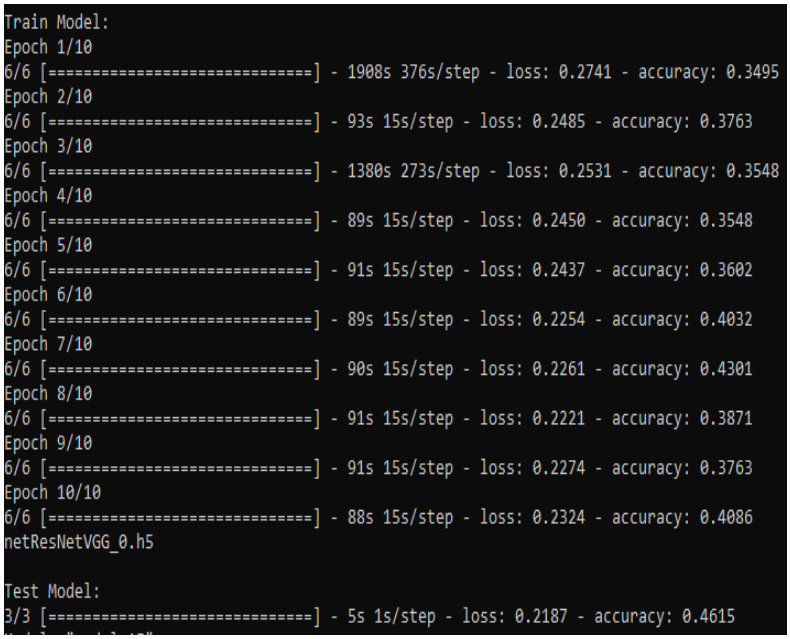

Рисунок 58. – ResNet + VGG-16 (Эксперимент 1)

Как видим обучение прошло до 41%, а на тестовых данных был показан результат в 46%, что показало результат лучше VGG-16 и хуже ResNet.

5) ResNet + MobileNet

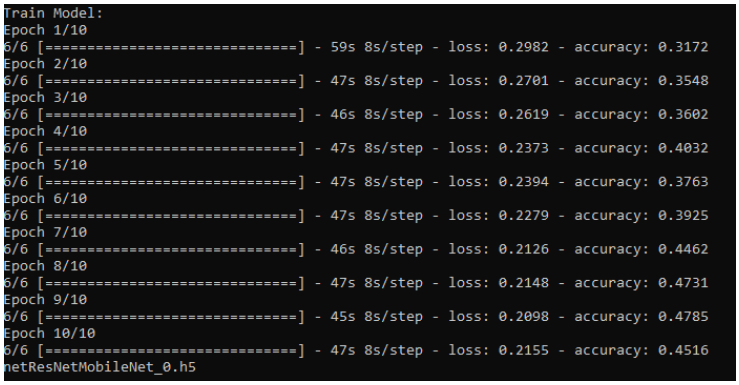

Рисунок 59. – ResNet + MobileNet (Эксперимент 1)

Как видим обучение прошло до 48%, а на тестовых данных был показан результат в 46%, что показало результат хуже MobileNet и хуже ResNet.

6) MobileNet + VGG-16

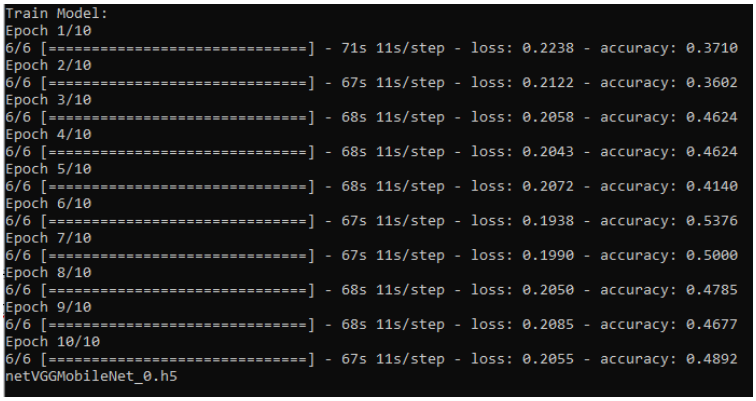

Рисунок 60. – MobileNet + VGG-16 (Эксперимент 1)

Как видим обучение прошло до 49%, а на тестовых данных был показан результат в 46%, что показало результат хуже MobileNet и лучше VGG-16.

7) MobileNet + VGG-16 + ResNet

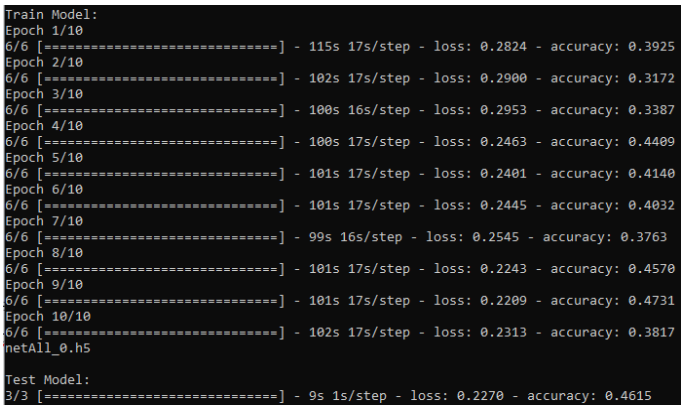

Рисунок 61. – MobileNet + VGG-16 + ResNet

Как видим обучение прошло до 38%, а на тестовых данных был показан результат в 46%, что показало результат всех комбинаций.

Таблица 1. – Результаты эксперимента 1

	train-loss	train-acc	test-loss	test-acc
ResNet	0.1762	0.5699	0.2239	0.4615
VGG-16	0.2219	0.3602	0.2216	0.4615
MobileNet	0.1651	0.5753	0.2169	0.4615
ResNet+VGG-16	0.2324	0.4086	0.2187	0.4615
ResNet+MobileNet	0.2155	0.4516	0.2206	0.4615
VGG-16+MobileNet	0.2055	0.4892	0.2202	0.4615
ResNet+VGG-16+MobileNet	0.2313	0.3817	0.2270	0.4615

3.3 Эксперимент 2

Во втором эксперименте была использована нейронная сеть с batch size = 16 и количеством эпох равным 20.

1) ResNet

```
Train Model:
Epoch 1/20
6/6 [==============================] - 40s 5s/step - loss: 0.2308 - accuracy: 0.3763
Epoch 2/20
6/6 [==============================] - 33s 6s/step - loss: 0.1977 - accuracy: 0.5108
Epoch 3/20
6/6 [==============================] - 33s 6s/step - loss: 0.2042 - accuracy: 0.4839
Epoch 4/20
6/6 [==============================] - 34s 6s/step - loss: 0.2023 - accuracy: 0.5108
Epoch 5/20
6/6 [==============================] - 34s 6s/step - loss: 0.1969 - accuracy: 0.5269
Epoch 6/20
6/6 [==============================] - 35s 6s/step - loss: 0.1922 - accuracy: 0.5376
Epoch 7/20
6/6 [==============================] - 34s 6s/step - loss: 0.1836 - accuracy: 0.5591
Epoch 8/20
6/6 [==============================] - 34s 6s/step - loss: 0.1886 - accuracy: 0.5484
Epoch 9/20
6/6 [==============================] - 34s 6s/step - loss: 0.1976 - accuracy: 0.5000
Epoch 10/20
6/6 [==============================] - 34s 6s/step - loss: 0.2032 - accuracy: 0.5215
Epoch 11/20
6/6 [==============================] - 34s 6s/step - loss: 0.1933 - accuracy: 0.5269
Epoch 12/20
6/6 [==============================] - 34s 6s/step - loss: 0.1834 - accuracy: 0.5269
Epoch 13/20
6/6 [==============================] - 34s 6s/step - loss: 0.1771 - accuracy: 0.5806
Epoch 14/20
6/6 [==============================] - 34s 6s/step - loss: 0.1723 - accuracy: 0.6290
Epoch 15/20
6/6 [==============================] - 34s 6s/step - loss: 0.1664 - accuracy: 0.6290
Epoch 16/20
6/6 [==============================] - 34s 6s/step - loss: 0.1559 - accuracy: 0.6720
Epoch 17/20
6/6 [==============================] - 34s 6s/step - loss: 0.1659 - accuracy: 0.5914
Epoch 18/20
6/6 [==============================] - 34s 6s/step - loss: 0.1501 - accuracy: 0.6505
Epoch 19/20
6/6 [==============================] - 34s 6s/step - loss: 0.1548 - accuracy: 0.6774
Epoch 20/20
6/6 [==============================] - 34s 6s/step - loss: 0.1455 - accuracy: 0.6505
```

Рисунок 62. – ResNet (Эксперимент 2)

Как видим обучение прошло до 65%, а на тестовых данных был показан результат в 46%.

2) VGG-16

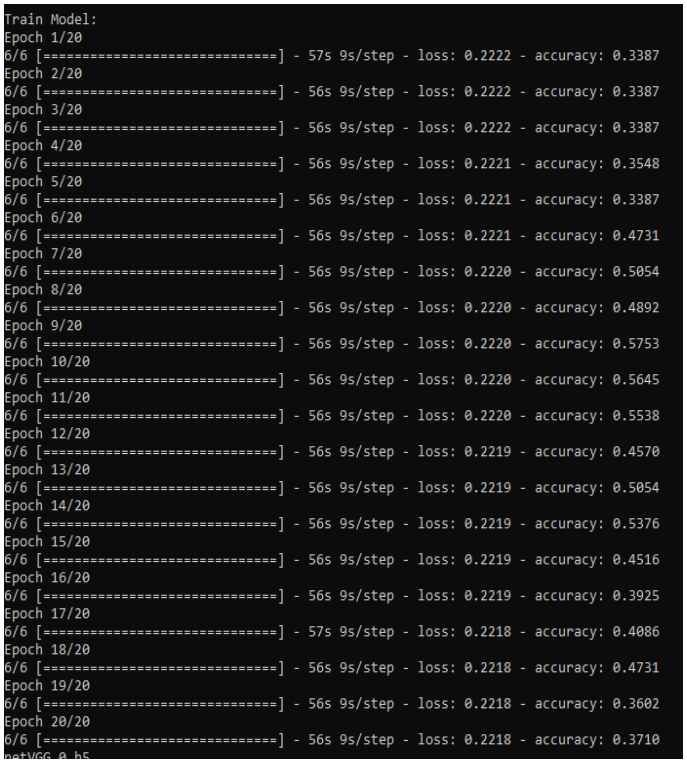

Рисунок 63. – VGG-16 (Эксперимент 2)

Как видим обучение прошло до 51%, а на тестовых данных был показан результат в 46%.

3) MobileNet

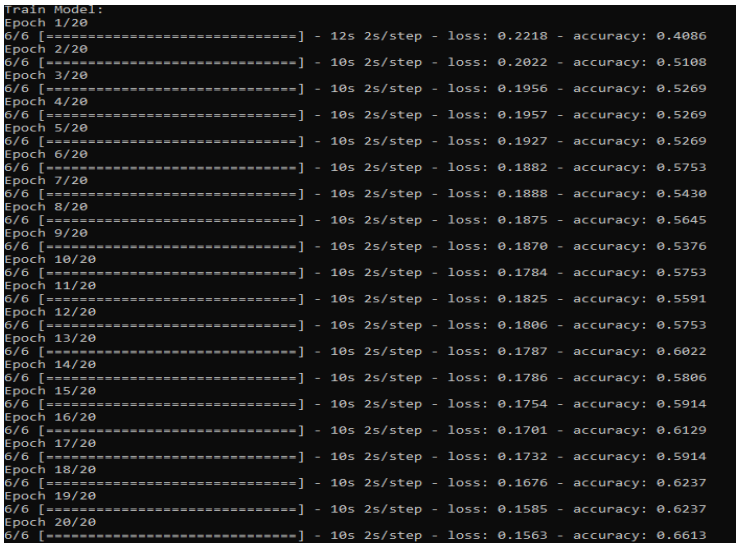

Рисунок 64. – MobileNet (Эксперимент 2)

Как видим обучение прошло до 66%, а на тестовых данных был показан результат в 23%.

4) ResNet + VGG-16

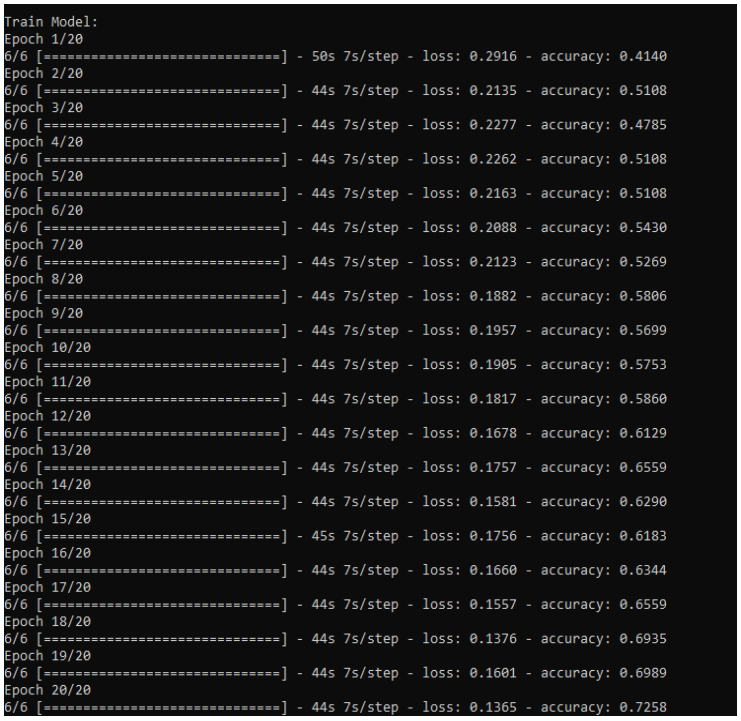

Рисунок 65. – ResNet + VGG-16 (Эксперимент 2)

Как видим обучение прошло до 73%, а на тестовых данных был показан результат в 46%, что показало результат лучше VGG-16 и лучше ResNet.

5) ResNet + MobileNet

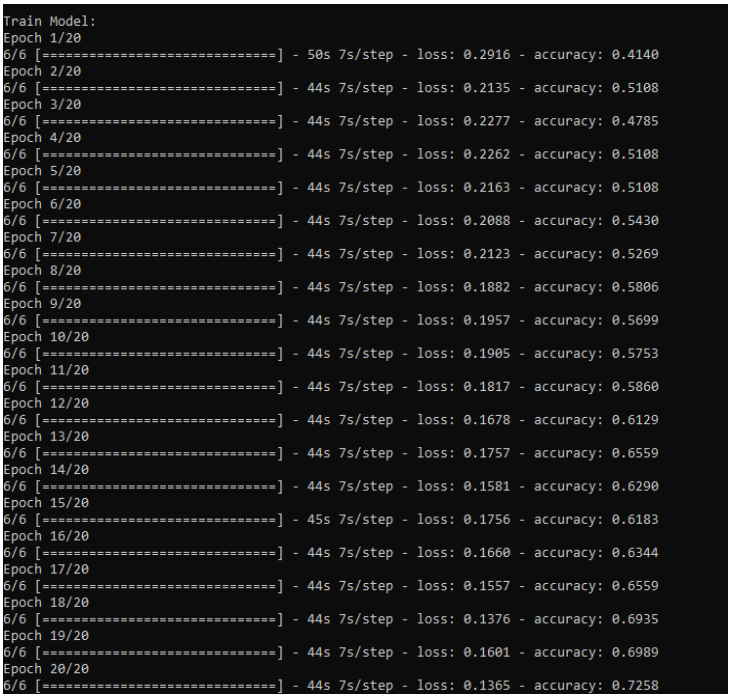

Рисунок 66. – ResNet + MobileNet (Эксперимент 2)

Как видим обучение прошло до 73%, а на тестовых данных был показан результат в 46%, что показало результат лучше MobileNet и лучше ResNet.

6) MobileNet + VGG-16

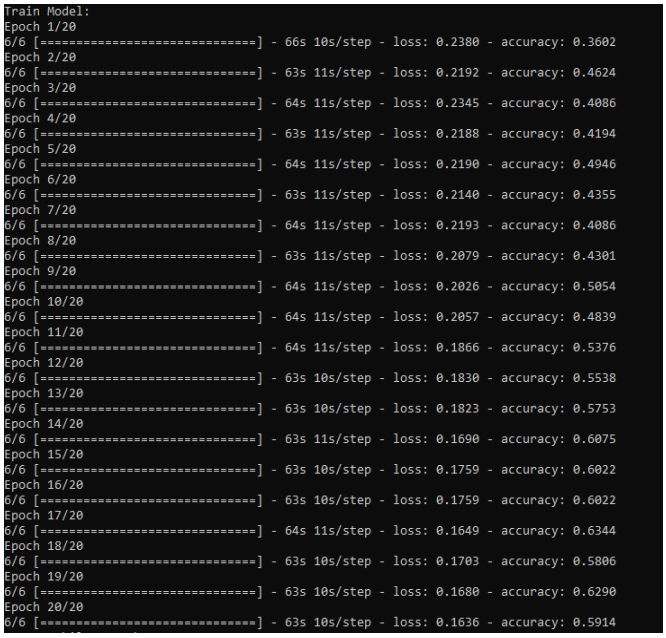

Рисунок 67. – MobileNet + VGG-16 (Эксперимент 2)

Как видим обучение прошло до 59%, а на тестовых данных был показан результат в 46%, что показало результат хуже VGG-16и лучше MobileNet.

7) MobileNet + VGG-16 + ResNet

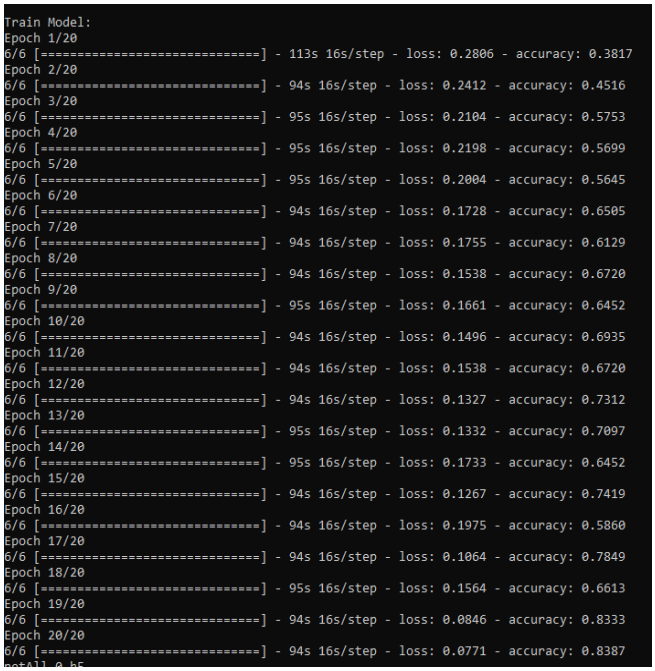

Рисунок 68. – MobileNet + VGG-16 + ResNet (Эксперимент 2)

Как видим обучение прошло до 84%, а на тестовых данных был показан результат в 54%, что показало результат всех комбинаций.

Таблица 2. – Результаты эксперимента 2

	train-loss	train-acc	test-loss	test-acc
ResNet	0.1455	0.6505	0.2503	0.4615
VGG-16	0.2218	0.3710	0.2212	0.4615
MobileNet	0.1563	0.6616	0.2308	0.4615
ResNet+VGG-16	0.1365	0.7258	0.2503	0.4615
ResNet+MobileNet	0.1365	0.7258	0.2746	0.4615
VGG-16+MobileNet	0.1636	0.5954	0.2394	0.4615
ResNet+VGG-16+MobileNet	0.0771	0.8387	0.2265	0.5385

3.4 Эксперимент 3

В третьем эксперименте была использована нейронная сеть с batch size = 12 и количеством эпох равным 25.

1) ResNet

```
Train Model:
Epoch 1/25
16/16 [==============================] - 47s 2s/step - loss: 0.2038 - accuracy: 0.4785
Epoch 2/25
16/16 [==============================] - 38s 2s/step - loss: 0.2016 - accuracy: 0.5054
Epoch 3/25
16/16 [==============================] - 38s 2s/step - loss: 0.1973 - accuracy: 0.5376
Epoch 4/25
16/16 [==============================] - 38s 2s/step - loss: 0.1912 - accuracy: 0.5806
Epoch 5/25
16/16 [==============================] - 38s 2s/step - loss: 0.2138 - accuracy: 0.4839
Epoch 6/25
16/16 [==============================] - 39s 2s/step - loss: 0.2141 - accuracy: 0.5269
Epoch 7/25
16/16 [==============================] - 39s 2s/step - loss: 0.1950 - accuracy: 0.5215
Epoch 8/25
16/16 [==============================] - 40s 2s/step - loss: 0.1928 - accuracy: 0.5323
Epoch 9/25
16/16 [==============================] - 38s 2s/step - loss: 0.1746 - accuracy: 0.6183
Epoch 10/25
16/16 [==============================] - 39s 2s/step - loss: 0.1779 - accuracy: 0.6022
Epoch 11/25
16/16 [==============================] - 37s 2s/step - loss: 0.1569 - accuracy: 0.7043
Epoch 12/25
16/16 [==============================] - 37s 2s/step - loss: 0.1515 - accuracy: 0.6935
Epoch 13/25
16/16 [==============================] - 37s 2s/step - loss: 0.1231 - accuracy: 0.7258
Epoch 14/25
16/16 [==============================] - 37s 2s/step - loss: 0.1116 - accuracy: 0.7742
Epoch 15/25
16/16 [==============================] - 37s 2s/step - loss: 0.1091 - accuracy: 0.7742
Epoch 16/25
16/16 [==============================] - 37s 2s/step - loss: 0.0976 - accuracy: 0.8011
Epoch 17/25
16/16 [==============================] - 38s 2s/step - loss: 0.1018 - accuracy: 0.7849
Epoch 18/25
16/16 [==============================] - 37s 2s/step - loss: 0.1015 - accuracy: 0.8065
Epoch 19/25
16/16 [==============================] - 37s 2s/step - loss: 0.0821 - accuracy: 0.8441
Epoch 20/25
16/16 [==============================] - 38s 2s/step - loss: 0.0831 - accuracy: 0.8495
Epoch 21/25
16/16 [==============================] - 37s 2s/step - loss: 0.0940 - accuracy: 0.8226
Epoch 22/25
16/16 [==============================] - 38s 2s/step - loss: 0.0932 - accuracy: 0.8172
Epoch 23/25
16/16 [==============================] - 41s 3s/step - loss: 0.0938 - accuracy: 0.8118
Epoch 24/25
16/16 [==============================] - 39s 2s/step - loss: 0.0746 - accuracy: 0.8495
Epoch 25/25
16/16 [==============================] - 39s 2s/step - loss: 0.0841 - accuracy: 0.8333
```

Рисунок 69. – ResNet (Эксперимент 3)

Как видим обучение прошло до 83%, а на тестовых данных был показан результат в 59%.

2) VGG-16

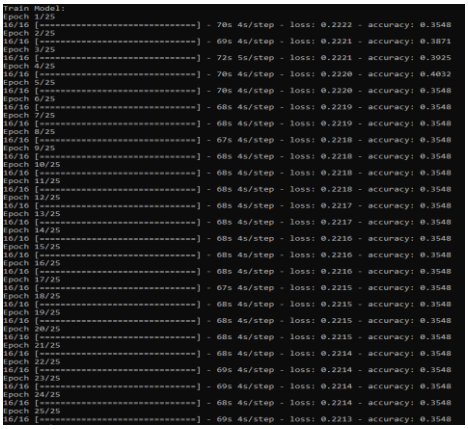

Рисунок 70. – VGG-16 (Эксперимент 3)

Как видим обучение прошло до 36%, а на тестовых данных был показан результат в 46%.

3) MobileNet

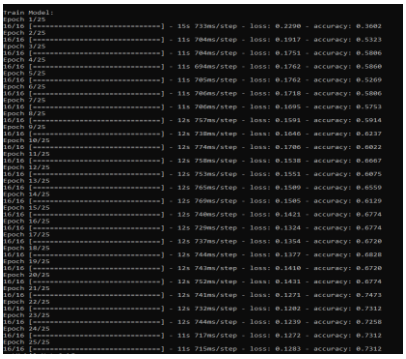

Рисунок 71. – MobileNet (Эксперимент 3)

Как видим обучение прошло до 73%, а на тестовых данных был показан результат в 46%.

4) ResNet + VGG-16

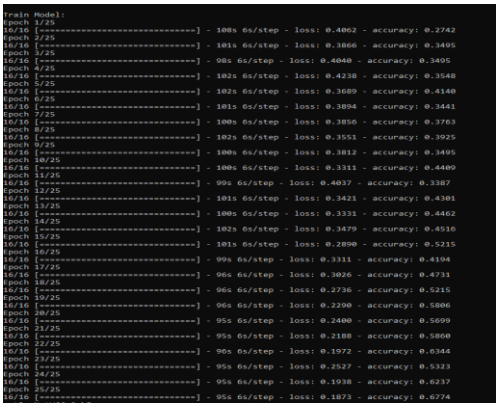

Рисунок 72. – ResNet + VGG-16 (Эксперимент 3)

Как видим обучение прошло до 68%, а на тестовых данных был показан результат в 49%.

5) ResNet + MobileNet

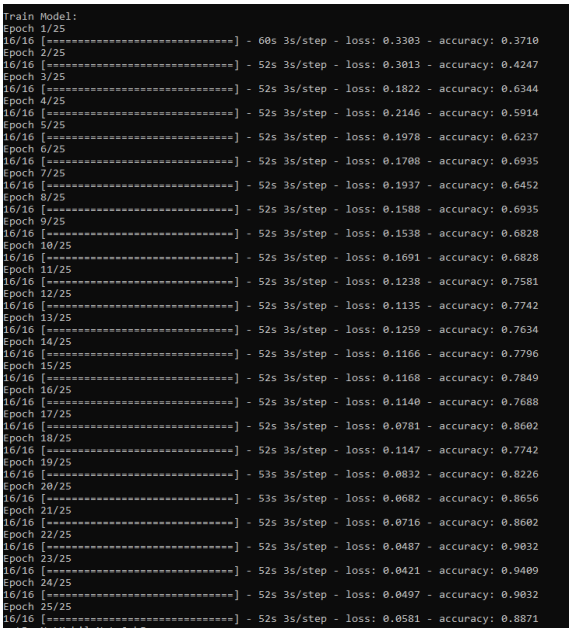

Рисунок 73. – ResNet + MobileNet (Эксперимент 3)

Как видим обучение прошло до 89%, а на тестовых данных был показан результат в 80%.

6) MobileNet + VGG-16

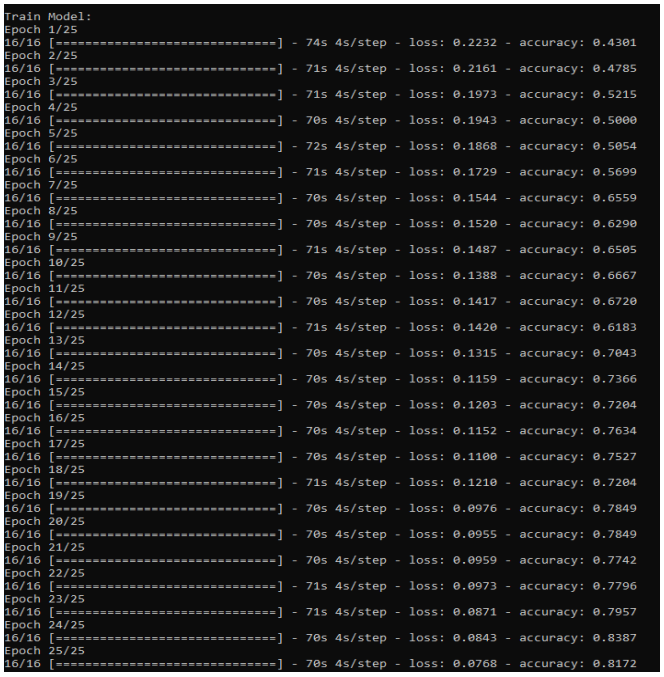

Рисунок 74. – MobileNet + VGG-16 (Эксперимент 3)

Как видим обучение прошло до 82%, а на тестовых данных был показан результат в 49%.

7) MobileNet + VGG-16 + ResNet

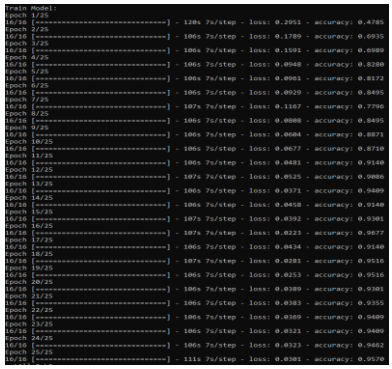

Рисунок 75. – MobileNet + VGG-16 + ResNet (Эксперимент 3)

Как видим обучение прошло до 96%, а на тестовых данных был показан результат в 90%, что показало результат всех комбинаций.

Таблица 3. – Результаты эксперимента 3

Model	train-loss	train-acc	test-loss	test-acc
ResNet	0.0841	0.8333	0.2190	0.5897
VGG-16	0.2213	0.3548	0.2197	0.4615
MobileNet	0.1283	0.7312	0.2403	0.4615
ResNet+VGG-16	0.1873	0.6774	0.2644	0.4872
ResNet+MobileNet	0.0581	0.8871	0.1125	0.7949
VGG-16+MobileNet	0.0768	0.8172	0.2861	0.4872
ResNet+VGG-16+MobileNet	0.0301	0.9570	0.0674	0.8974

Пример:

Рисунок 76. – Alia Bhat

Таблица 4. – Результаты тестов

Нейронные сети	Эксперимент 1	Эксперимент 2	Эксперимент 3
ResNet	+	-	-
VGG-16	+	+	+
MobileNet	-	-	-
ResNet + VGG-16	-	-	+
ResNet + MobileNet	-	-	-
MobileNet + VGG-16	-	-	+
ResNet + VGG-16 + MobileNet	-	+	+

Как вилим в примере, ансамбль сетей показал лучшие результаты, чем нейронные сети по отдельности

3.4 Вывод к разделу 3.

Из экспериментов, представленных в этой главе мы можем сделать вывод о том, что ансамбли нейронных сетей действительно могут улучшать результат обучения нейронной сети. Но также можем видеть, что при неправильно подобранных гиперпараметров и обучении на малых дистанциях ансамбль не улучшает обучение, а в некоторых случаях и ухудшает его. Поэтому для ансамблей надо использовать больше эпох, так как их точность раскрывается позднее.

Заключение

В ходе работы были исследованы существующие методы распознавания лиц. После изучения существующих методов, были предложены свои алгоритмы для решения задачи распознания лиц, а именно использование ансамблирования таких нейронных сетей, как VGG-16, ResNet и MobileNet. Также были проведены вычислительные эксперимента распознавани сетей VGG-16, ResNet и MobileNet, а также их вариации ансамблей. В самих экспериментах мы можем увидеть, что ансамблирование этих архитектур показало увеличение точности обучения и распознавания. Но для достижения увеличения точности необходимо большее количество эпох, что и показывают нам эксперименты.

Список литературы

1) C. Papageorgiou, M. Oren and T. Poggio. A General Framework for Object Detection. *International Conference on Computer Vision*, 1998.

2) Брилюк Д.В., Старовойтов В.В. Распознавание человека по изображению лица нейросетевыми методами. – Минск, 2002. – 54 с.

3) Распознавание лиц: понимание алгоритма LBPH [Электронные ресурс] https://www.machinelearningmastery.ru/face-recognition-how-lbph-works-90ec258c3d6b/

4) Распознавание лиц для начинающих [Электронные ресурс] https://www.machinelearningmastery.ru/face-recognition-for-beginners-a7a9bd5eb5c2/

5) Метод главных компонент (PCA) [Электронные ресурс] https://synset.com/ai/ru/recognition/Recognition_PCA.html

6) Ансамбли нейронных сетей [Электронный ресурс] http://www.100byte.ru/python/twoNets/twoNets.html

7) Распознавание изображений. Алгоритм Eigenface [Электронный ресурс] https://habr.com/ru/post/68870/

8) Распознавание лиц с помощью сиамских сетей [Электронный ресурс] https://habr.com/ru/companies/jetinfosystems/articles/465279/

9) Распознаем эмоции на лице [Электронный ресурс] https://habr.com/ru/companies/otus/articles/722284/

10) Распознавание лиц с InsightFace [Электронный ресурс] https://habr.com/ru/articles/699232/

11) Распознавание лиц с FaceNet и MTCNN [Электронный ресурс] https://questu.ru/articles/73163/

12) Что такое Mediapipe [Электронный ресурс] https://habr.com/ru/companies/agima/articles/696836/