МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

«Бизнес-информатика»
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного комплекса по определению и стиранию очков с лица человека»

Обучающийся Е.О. Прохорчик
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Казаченок
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Бакалаврская работа выполнена на тему «Разработка программного комплекса по определению и стиранию очков с лица человека».

Цель работы состоит в разработке программного комплекса с использованием нейронных сетей для определения и стирания очков с лица человека.

Во введении определены: актуальность темы, объект и предмет исследования, методы исследования, цель работы, а также задачи для достижения поставленной цели.

В первой главе проведен анализ компании и отдела, для которого выполнялась разработка, построены и описаны модели бизнес-процессов, сформулированы решаемые задачи.

Во второй главе определены и описаны этапы разработки, выбраны методы решения. Даны теоретические выкладки и описана предметная область для решаемой задачи.

В третьей главе описан процесс тестирования, приводятся результаты работы и возможные варианты улучшения.

В заключении приводятся выводы и по итогам разработки комплекса.

Бакалаврская работа состоит из 50 страниц и включает 33 рисунка, 20 источников.

Оглавление

Введение

Данная работа рассматривает применение алгоритмов машинного обучения и нейронных сетей в решении реальной практической задачи. Основой для работы послужило моё обучение на должность инженера машинного обучения. Заказчик – один из крупнейших англоязычных производитель очков и контактных линз.

Цель работы – разработка программного комплекса по определению и стиранию очков с лица человека.

Объект исследования – методы обнаружения и удаления объектов с изображения.

Предметом исследования является применение искусственных нейронных сетей для обнаружения и удаления объектов.

Основными методами являются искусственные нейронные сети и алгоритмы компьютерного зрения.

В ходе написания работы были решены следующие задачи:

– анализ и проектирование предметной области;
– выбор архитектуры нейронных сетей;
– выбор моделей, основанных на архитектуре;
– сбор и разметка наборов данных;
– обучение моделей;
– написание кода для использования моделей
– тестирование результатов

Работа является актуальной, поскольку в нынешнее время популярность и доступность машинного обучения и его областей растёт, и многие компании задумываются и внедряют в свою работу решения, использующие методы машинного обучения.

Глава 1. Анализ деятельности компании «Optical Outfitters»

1.1 Характеристика компании

Основной деятельностью американской компании «Optical Outfitters» является производство и продажа товаров, связанных со зрением (очки, контактные линзы, расходные материалы). Магазины компании насчитывают более 60 штук и представлены более чем в 15 штатах.

Компания является одной из самых популярных компаний в сфере здоровья глаз. Причины этого в следующем:

– наличие приложения для удаленной примерки очков;
– возможность бесплатного заказа нескольких оправ для примерки;
– наличие современного оборудования и квалифицированного персонала для проверки зрения практически во всех розничных магазинах;
– возможность отдельного заказа проверки зрения.

Так как компания имеет множество офисов по всем Соединенным Штатам, то управление разбито по регионам. Структура компании на верхнем уровне представлена на рисунке 1.

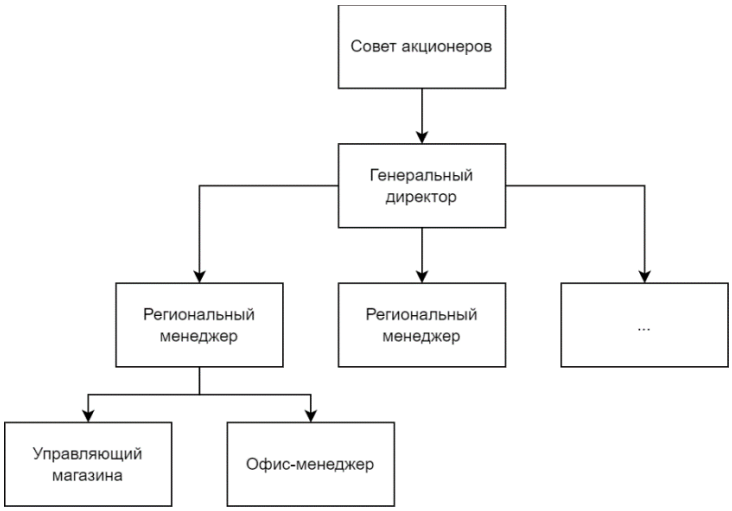

Рисунок 1 – Верхний уровень управления компании «Optical Outfitters»

Структура компании представляет собой классическую схему крупной компании с большим числом офисов по стране. Регион представляет собой один или несколько штатов, в зависимости от их размера. Для каждого региона назначен свой менеджер, которым подчиняются магазины и офисы в регионе.

1.2 Описание розничного магазина

Так как комплекс разрабатывается для облегчения работы розничных магазинов, то целесообразно будет дать описание структуры магазина и бизнес-процессов, которые будут изменены комплексом.

Структура магазина представлена на рисунке 2.

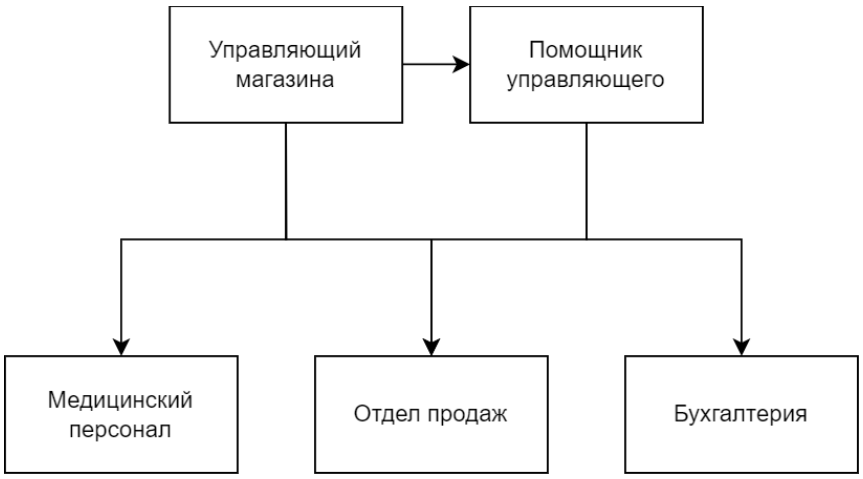

Рисунок 2 – Структура розничного магазина

Магазин состоит из следующих помещений: торгового зала, зала для проверки зрения, рабочего места бухгалтерии, комнат для персонала, складского помещения.

Торговый зал оснащен витринами для и стойками для очков и разделен на секции по типу продукции. Имеется рабочее место продавца, которое оснащено отдельными шкафами для линз и расходных материалов.

Зала для проверки зрения состоит из двух помещений. В одном из них проводятся стандартные тесты на остроту зрения и проводятся консультации с офтальмологом. Другое помещение используется для проверки зрения на специальных аппаратах.

Все рабочие места оснащены компьютерами и принтерами, у продавцов имеются планшеты для записей и оформления заказов.

1.3 Моделирование процесса покупки оправы, анализ модели «Как есть»

В качестве технологии проектирования выбран графический стандарт IDEF0.

IDEF0 – является функциональной моделью, образующей ядро выстраивания всего остального комплекса, в ней связываются организационная структура, материальные потоки, информационные потоки, сама деятельность компании и управляющие воздействия, формируя единую конструкцию. Также можно называть нотацией графический стандарт, принятый для моделирования процессов. Таким образом, нотация является системой требований и правил, по которым должна строиться та или иная модель деятельности.

Нотация IDEF0 представляет собой достаточно строгую методику, разработанную в первую очередь, чтобы осуществлять ручное моделирование, аналогично стандартам технического конструирования. Это обуславливает то, что в ней есть требования, как размещать стрелки, какой формат элементов должен быть, каково содержание диаграммы информационной рамки к IDEF0 и т.д. Учитывая, что деятельность компании представляет собой сложную многоуровневую систему действий, в основном, получения много различных схем. Это предполагает, что необходимо делать систематизацию и разрабатывать навигацию в отношении всех элементов модели. В данный

момент этим занимаются, как правило, компьютерные системы, которые поддерживают построение моделей, используя данную нотацию.

Основным процессом, на котором будет сосредоточено внимание является процесс покупки оправы клиентом магазина. Покупка может производиться как в оффлайн режиме, когда клиент на месте выбирает оправы и делает заказ, так и онлайн с помощью специального приложения. На рисунке 3 представлена диаграмма процесса покупки очков «Как есть», а на рисунке 4 – его декомпозиция.

Рисунок 3 – Контекстная диаграмма процесса «Покупка оправы», модель «Как есть»

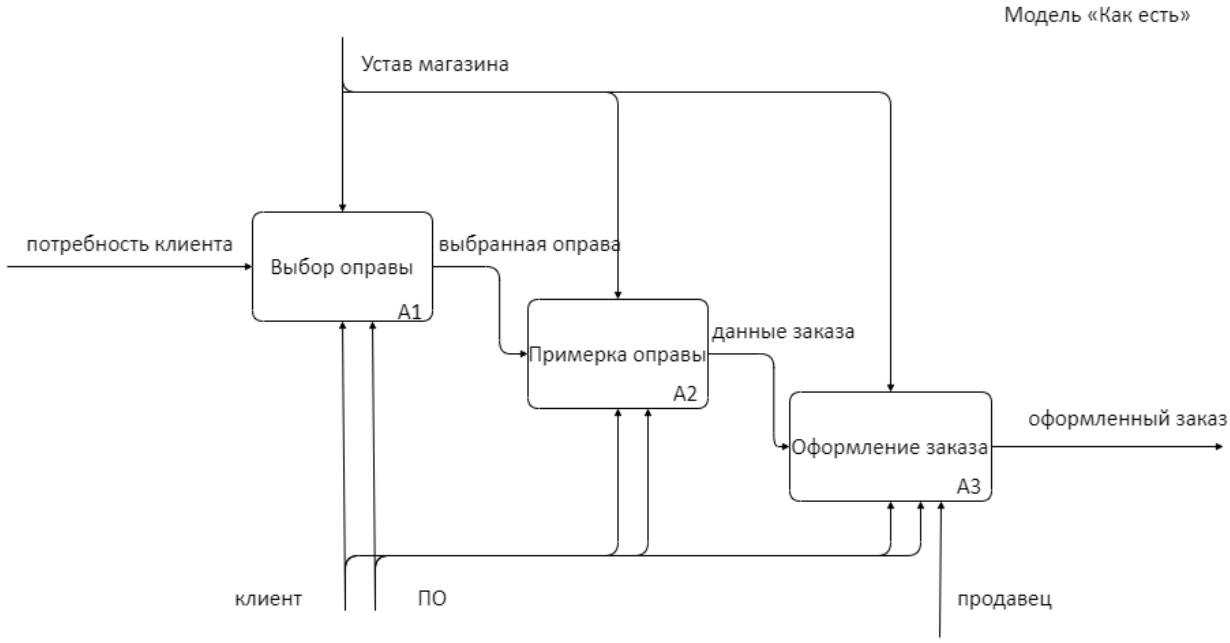

Рисунок 4 – Декомпозиция процесса «Покупка оправы»

Процесс состоит из 3 этапов: выбора оправы, примерка оправы, оформление заказа. Входом является потребность клиента в оправе.

На каждом этапе, в зависимости от того, оффлайн или онлайн покупка, один из исполнителей является либо продавцом, либо приложением. Выходом является оформленный заказ.

Разберем более подробно процесс примерки с использованием приложения, так как именно приложение необходимо доработать. Диаграмма процесса примерки представлена на рисунке 5.

Рисунок 5 – Декомпозиция процесса примерки оправы

Процесс состоит из трех этапов: сначала необходимо снять свои очки; затем приложение генерирует виртуальную модель выбранной оправы; после чего оправа накладывается на видео с фронтальной камеры, если на нем есть лицо, и клиент может под разными углами оценить посадку оправы.

1.4 Формулировка задачи на разработку

Проблемным местом является этап снятия очков A2.1. Люди сильной степенью близорукости или дальнозоркостью, которые не носят контактные линзы, не могут позволить себе снять очки, чтобы использовать приложение, потому что при очень сильном приближении камеры к лицу оправа не будет генерироваться. А при наложении виртуальной на реальную оправу очень трудно различить виртуальную.

Соответственно, требуется разработать некий алгоритм по удалению реальных очков с лица. Требования к алгоритму:

– частота кадров после обработки не менее 24 кадров в секунду;

– работа как на темных контрастных оправах, так и на светлых;
– независимость от цвета кожи и возраста;
– независимость от наличия маски на лице.

Алгоритм необходимо будет встроить в существующее приложение, поэтому должны быть возможность переноса кода на другой язык программирования и широкая поддержка используемых методов и библиотек.

1.5 Моделирование диаграммы процесса покупки «Как должно быть»

После формулировки задачи перейдем к моделированию диаграмм «Как должно быть». Доработанное алгоритмом ПО будет непосредственно влиять на процесс покупки оправы с помощью приложения. Контекстная диаграмма процесса «Покупка оправы» представлена на рисунке 6.

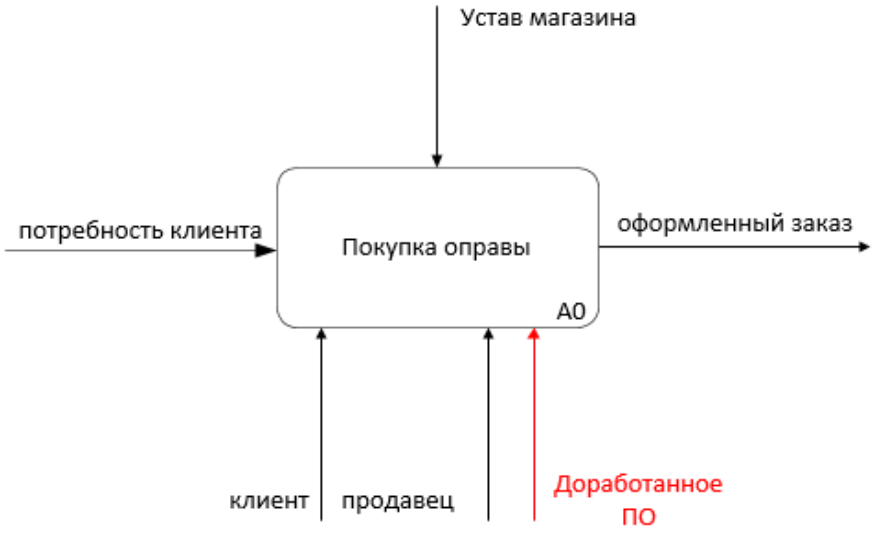

Рисунок 6 – Контекстная диаграмма процесса «Покупка оправы», модель «Как должно быть»

Доработанное ПО по-прежнему участвует в трех этапах покупки оправы, что изображено на рисунке 7.

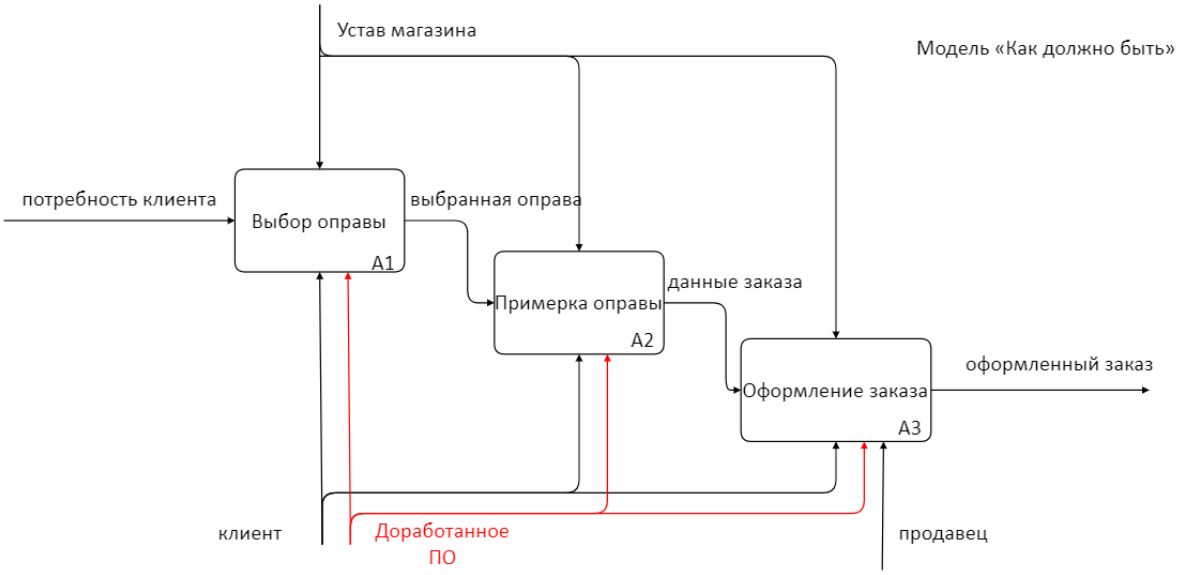

Рисунок 7 – Декомпозиция процесса «Покупка оправы», модель «Как должно быть»

Однако, основное изменение происходит на этапе примерки оправы. Этапы выбора и оформления заказа остаются без изменений, что изображено на рисунке 8.

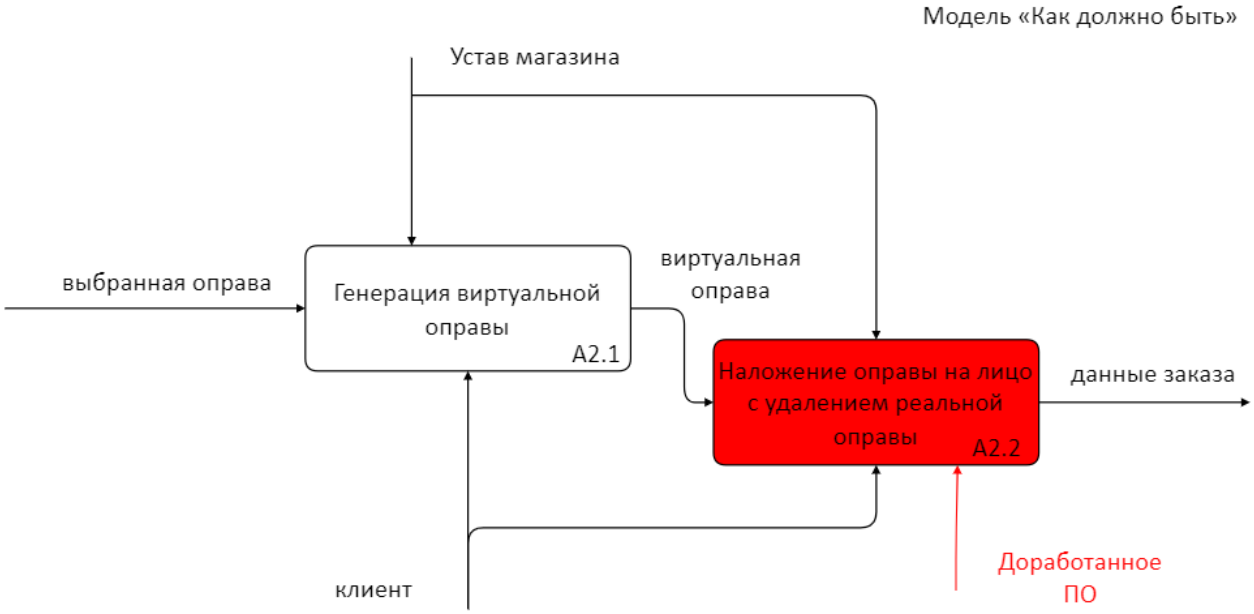

Рисунок 8 – Декомпозиция процесса примерки оправы, модель «Как должно быть»

После доработки приложения пользователь сможет примерять виртуальные оправы, не снимая собственных очков. Таким образом приложением смогут пользоваться большее число людей.

Выводы и результаты по главе 1

В данной главе был проведен анализ деятельности компании «Optical Outfitters» В ходе анализа бизнес-процессов и диаграмм «Как есть» процесса покупки очков клиентом с помощью приложения был выявлен проблемный этап в примерке очков. Исходя из него была сформулирована задача для разработки алгоритма доработки и определены его основные требования.

Глава 2. Разработка программного комплекса

2.1 Определение и этапов жизненного цикла и этапов разработки программного комплекса, формулировка задач в контексте машинного обучения

2.1.1 Жизненный цикл продукта

У любого программного продукта есть свой жизненный цикл, определяющий этапы, через которые проходит продукт от идеи до финального результата. Программный комплекс в этой дипломной работе имеет следующие этапы женненного цикла:

- идея;
- анализ области и составления требований;
- проектирование;
- разработка;
- тестирование.

Этап внедрения отсутствует в силу того, что основная цель этого комплекса – создание работающего алгоритма, на котором будет проводится тестирование в различных условиях для выявления ограничений по применению.

Основная идея – создать некую программу, которая сможет удалять с изображений или видео различные очки, и использовать при этом нейронные сети. Следовательно, для начала необходимо понять, что вообще такое нейронные сети, как они работают, а затем выбрать подходящие варианты. А поскольку нейронные сети – это часть машинного обучения, то так же надо дать определение и этому понятию.

2.1.2 Этапы разработки и описание задач

Поскольку программный комплекс предполагает использование нейронных сетей, то этапы разработки будут несколько отличатся от классических программ.

Процесс разработки состоит из четырех этапов:

- выбор архитектуры нейронной сети;
- выбор моделей нейронных сетей, основанных на выбранной архитектуре. Этот этап влияет на формат данных и разметки, определяет программные платформы для обучения;
- составление и разметка наборов данных. Самый трудоемкий и длительный этап;
- обучение моделей;
- написание кода на Python для обработки изображений обученными сетями.

В контексте нейронных сетей задачи выглядят следующим образом: найти на видео очки, определить пиксели, которые являются очками и закрасить их максимально незаметно. Первая часть относится к области обнаружения объектов, вторая – к семантической сегментации, за третью часть будет отвечать классический алгорритм компьютерного зрения.

Обнаружение (детекция) объектов (англ. object detection) – задача, при которой на изображении необходимо выделить прямоугольную область, в которой находится объект. Объект может находится в любом месте изображеия и даже частично выходить за его границу. [8], [16]

При семантической сегментации (англ. semantic segmentation) задача состоит в том, чтобы найти и выделить пиксели, принадлежащие объекту. Как и в предыдущей задаче, положение объекта на изображении может быть произвольным. [11], [16]

Обе эти задачи применимы к множетсвенному числу и типу объектов. Сущетсвует вариация семанитческой сегментации под названием сегментация экземпляров. В отличие от семантической, данный вариант рассматривает разные виды объектов как разные классы. Например, если на картинке три человека, то при семантической сегментации все пиксели людей будут помечены общим классом «человек», а при сегментации экземпляров, каждый

человек рассматривается как уникальный экземпляр своего класса. На рисунке 9 изображен пример решения двух данных задач.

Рисунок 9 – примеры детекции и сегментации

Закрашивание (англ. inpainting) – это задача восстановления отсутствующих частей изображения [13]. На рисунке 10 – пример закрашивания.

Рисунок 10 – Пример закрашивания

Для решения воспользуемся классическим алгоритмом, так как он показал неплохие результаты в данном случае. Классический значит, что он придуман без использования нейронных сетей. По сути своей это

математические операции над значениями пикселей изображения [1]. Плюсами таких алгоритмов являются скорость работы и простота использования. Часто они представлены готовыми функциями в библиотеках для работы с изображениями. Из минусов – низкая устойчивость результатов к различным изображениям.

2.2 Машинное обучение, глубокое обучение, нейронные сети

Машинное обучение (англ. Machine Learning) – это область искусственного интеллекта, алгоритмы которой решают задачу не прямым способом, а с помощью обучения на похожих задачах [2]. Обучение – это процесс нахождения скрытой зависимости в известных данных, которая позволит предсказывать ответы на новых, ранее не доступных данных [8]. Общий набор данных для задачи называется датасетом**.** Вид датасета зависит от типа задачи, например табличные данные, набор слов, изображения или видео, звуковые дорожки. Обучение же происходит на подвыборке из датасета, которая называется тренировочной выборкой (англ. training set).

Глубокое обучение (англ. Deep Learning) – это подраздел машинного обучение, использующий в качестве алгоритмов нейронные сети. Для обучение таких алгоритмов нужно гораздо больше данных, чем для классических алгоритмов машинного обучения.

Нейронная сеть или искусственная нейронная сеть (ИНС) (англ. Artificial Neural Networks, ANN) – математическая модель, вдохновленная человеческим мозгом и работой его нейронов. Нейронные сети – это «сердце» глубокого обучения.

Несмотря на то, что понятие ИНС известно еще с 1940-х годов, массовость они стали приобретать на волне очередных исследований в этой области в начале 2000-х. Поэтому сейчас нейронные сети могут решать достаточно сложные задачи (например классификация и генерация изображений, текста и т.д.).

Рассмотрим пример одной из простейших архитектур ИНС для понимания работы искусственных нейронов [6].

На рисунке 11 изображён однослойный перцептрон

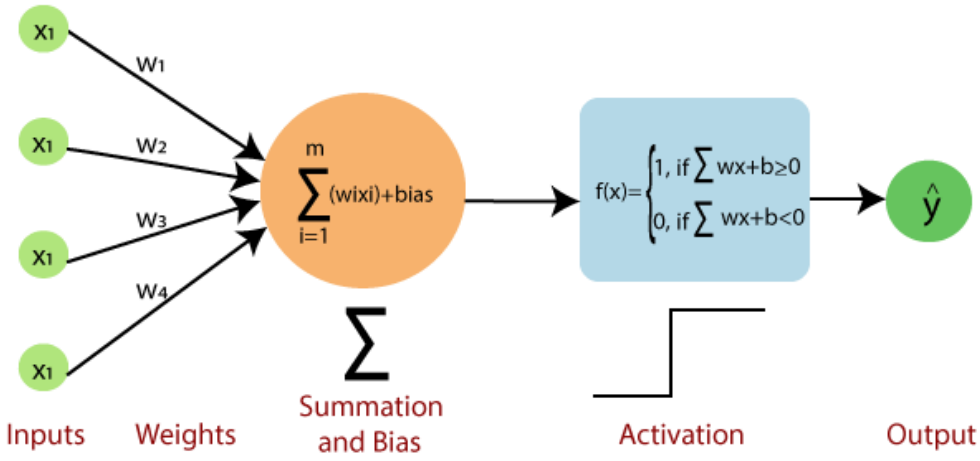

Рисунок 11 – Схема однослойного перцептрона

Принцип работы:

– на вход подается вектор из чисел. Входные значения называются входным слоем;
– это признаки из тренировочной выборки;
– каждый признак из входного вектора перемножается с числом, называемым весом;
– произведения из предыдущего шага складываются, к ним добавляется константа смещения;
– эта сумма проходит через функцию активации;
– полученное значение есть результат работы перцептрона.

Нейронная сеть в общем случае состоит из большого числа нейронов (или узлов), объединенных в слои, где нейроны каждого слоя соединены с нейронами предыдущего и последующего слоя. Чаще всего у сети один входной и один выходной слой, который может содержать, в зависимости от задачи, от одного до нескольких выходных значений.

Функция активации определяет значение выхода нейронной сети. Обычно данная функция является нелинейной, что позволяет нейронной сети строить сложные разделяющие поверхности и делать нелинейные приближения.

Вид сети может сильно отличатся в зависимости от задачи. Структура сети (количество и тип слоев, количество и тип нейронов, вид функции активации и т.д.) определяется архитектурой нейронной сети.

На рисунке 12 показана сеть с двумя скрытыми слоями и двумя выходами.

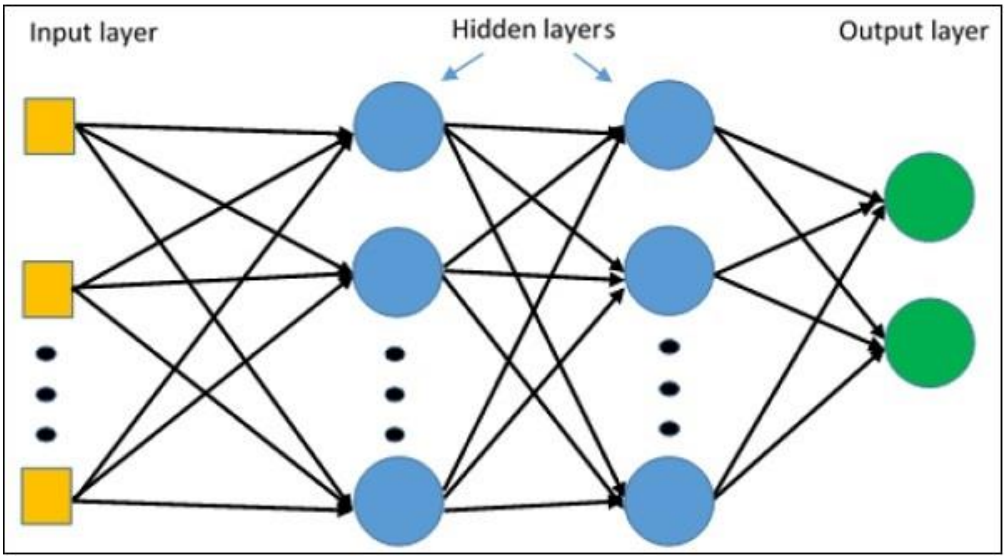

Рисунок 12 – Схема сети с двумя скрытыми слоями

Для того, чтобы понимать, насколько точные ответы даёт сеть, и сравнивать результаты нейронной сети с реальными значениями, заранее определяется специальная функция под названием функция потерь (ФП) (англ. Loss function). Одна из самых распространённых ФП – среднеквадратичная ошибка (англ. Mean Squared Error, MSE). Она представляет собой среднюю сумму квадратов разности предсказанного и реального значений.

2.3 Метод обратного распространения ошибки для обучения нейронных сетей

Обучение нейронной сети состоит в том, чтобы минимизировать значение функции потерь. Для этого используется метод обратного распространения ошибки (англ. Backpropagation), Изобретение которого дало новый толчок к изучению нейронных сетей. Данный метод позволяет автоматически обновлять веса нейронной сети, используя градиентный спуск (англ. gradient descent) и правило цепочки математического анализа для подсчёта производных сложных функций [9].

Градиентный спуск позволяет итерационным способом из случайной точки на функции найти минимум этой функции.

Метод обратного распространения ошибки состоит из двух этапов:

- сначала в каждом слое сети для каждого нейрона считается его выход, который передается в последующие слои. Так происходит пока не будет достигнут выходной слой. Это так называемый прямой проход (англ. forward pass);
- затем считается значение функции потерь и в обратном порядке считаются производные для каждого нейрона в каждом слое, вплоть до входного слоя. Затем веса корректируются с помощью выполнения итерации градиентного спуска. Это обратный проход (англ. backward pass).

Этот метод является основным в глубоком обучении.

2.5 Выбор архитектуры

Так как изображения – это сложный многомерный объект (чаще всего это трехмерная матрица), то для возможности работы с ними были придуманы специальные архитектуры.

Свёрточная нейронная сети (англ. Convolutional neural network) – архитектура, вдохновленная зрительной корой человеческого мозга, которая, в отличие от полносвязных использует специальные слои, а именно свёрточный слой и слой пулинга (подвыборки) [3], [10].

Свёрточный слой производит над изображением операцию свёртки с использованием так называемых ядер. Ядро — это матрица меньшей размерности, состоящая из фильтров, также матриц. Количество фильтров в ядре зависит от количества каналов в изображении. Обычно, изображения подаются с трема каналами: R (red), G (green), B (blue). На рисунке 13 изображён пример свертки [5].

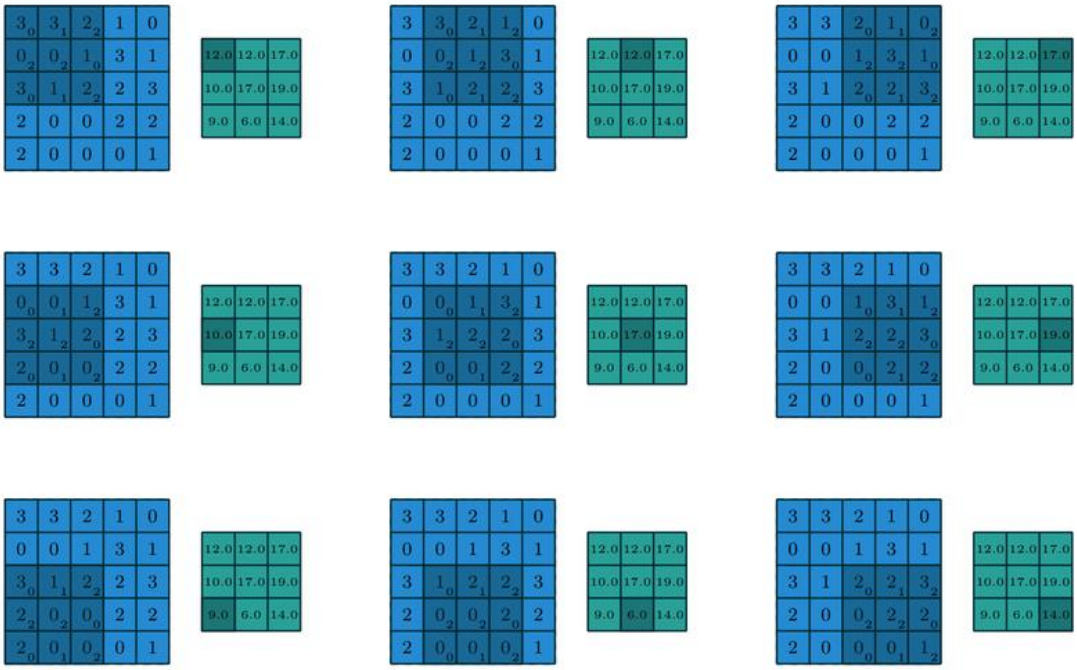

Рисунок 13 – Пример свёртки 3x3 для матрицы 5x5

Сверточный слой имеет два важных параметра: размер шага (англ. stride) и отступ (англ. padding). Размер шага влияет на расстояние, на которое перемещается ядро свертки. В приведенном выше примере размер шага равен 1.

Отступ используется, если необходимо, чтобы размер выхода после свёртки был равен размеру входа. Обычно картинка расширяется нулями. На рисунке 14 изображена свёртка 3x3 с применением нулевого отступа. Как видно, размер после свёртки равен 5x6, что совпадает с размером до свёртки [4].

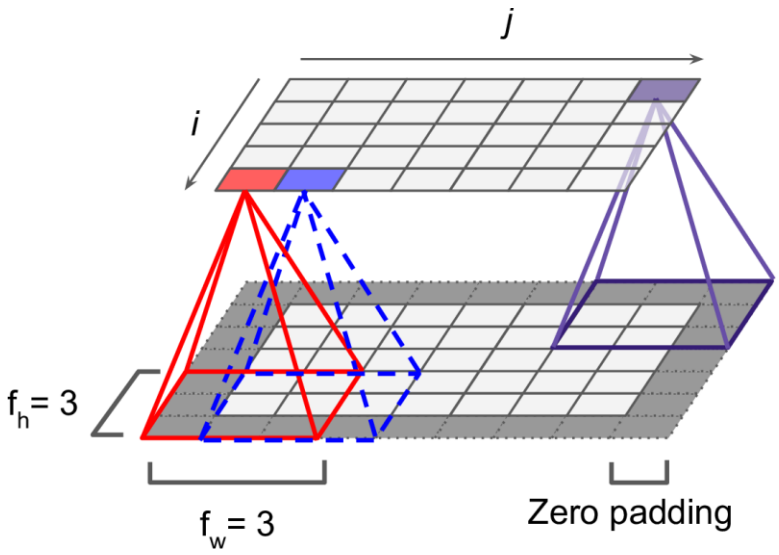

Рисунок 14 – Пример нулевого отступа

Ядра позволяют выделять из изображения признаки: как низкоуровневые, вроде линий и окружностей, так и высокоуровневые, например части объектов или сами объекты целиком Размер ядер и их количество выбираются до начала обучения. Также стоит отметить, что операция свёртки позволяет не только выделять признаки, но и делает это вне зависимости от относительного расположения объекта на изображении.

Слой пулинга в первую очередь используется для понижения размерности изображения. Для этого к некой заранее выбранной области применятся некая функция (зачастую это среднее или максимум) (рисунок 15).

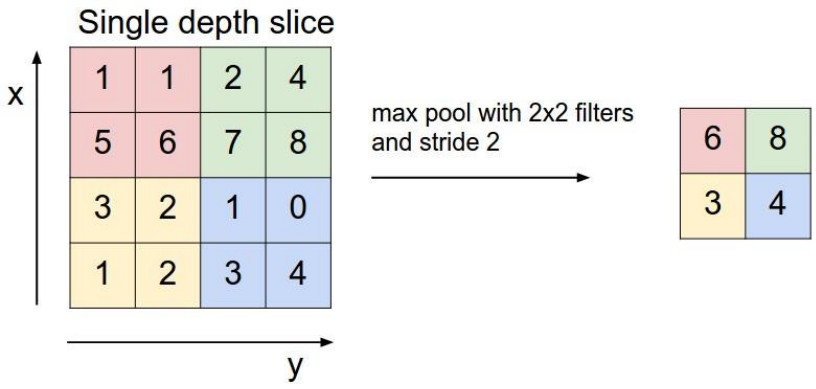

Рисунок 15 – Пример подвыборки 2x2 с взятием максимума

Снижение размерности ведет к уменьшения числа операций и ускорения работы сети [5].

Таким образом свёрточные нейронные сети с их возможностью нарождения ключевых признаков вне зависимости от их положения являются лучшим выбором для работы с изображениями.

2.6 Выбор моделей с архитектурой сверточной нейронной сети

Первой идеей было использовать какую-то одну модель, которая могла бы как определять регион с очками, так и производить его сегментацию. От этой идеи было решено отказаться, так как задачи требуют запоминания слишком различных признаков, модель вышла бы большой и громоздкой, что негативно сказалось бы на скорости работы. Поэтому было решено для задач, требующих применения нейронных сетей, использовать различные заранее предобученные модели. Использование готовых обученных моделей называется транзитным обучением (англ. transfer learning).

Транзитное обучение – подход при котором, используется заранее предобученная на большом датасете, например ImageNet, модель. На таком датасете модели учатся доставать и запоминать признаки [12]. Для того, чтобы обнаруживать любые пользовательские классы объектов, обучению подвергаются только последние слои.

Причина такого решения в том, что для составления собственной рабочей архитектуры необходимо затратить огромное количество времени как для изучения математических принципов работы нейронных сетей, свёрточных слоев и т.д., так и для подбора нужных слоев в нужном количестве и последовательности. Кроме этого, для первоначального обучения понадобилось бы гораздо большее количество размеченных данных, чем мы могли себе позволить собрать. Например, один из самых популярных датасетов с изображениями под названием ImageNet, на котором обучаются и проходят тесты многие модели, в том числе и выбранные, содержит более 14 миллионов вручную размеченных изображений.

2.6.1 Модель для обнаружения объектов YOLO

Начнем с модели для обнаружения региона с очками.

После изучения связанных с этой темой статей и материалов было решено использовать готовую предобученную под названием YOLO (You Only Look Once), а точнее его облегченную 4 версию YOLOv4 tiny [7].

YOLO – это популярный алгоритм для обнаружения объектов в режиме реального времени. С помощью этого алгоритма процесс классификации и прогнозирования ограничивающих рамок (англ. bounding boxes) для обнаруженных объектов, который раньше был многоступенчатым и требовал несколько сетей, теперь выполняется, используя одну нейронную сеть. Это позволило оптимизировать и значительно ускорить весь процесс. Кроме того, был пересмотрен подход к задаче определения ограничивающих рамок. Описание работы я приведу для алгоритма первой версии, YOLOv1, потому что основная идея, лежащая в основе архитектуры, остается неизменной. Назван алгоритм You Only Look Once потому, что он обучается на всем изображении целиком, в отличие от других архитектур, которые рассматривают лишь части изображений. Это позволят сети запоминать контекст и обобщать признаки объекта, что даёт независимость от положения объекта на изображении и делает сеть более применимой к различным новым средам. Теперь рассмотрим подробнее, как она работает.

Алгоритм основан на идее разбиения изображения на квадратную сетку размером SxS, как на рисунке 16 ниже:

Рисунок 16 – Пример сегментации изображения на ячейки

Ячейка, в которой находится центр объекта, например, центр собаки, является ответственной за обнаружение этого объекта. Каждая ячейка будет предсказывать число B ограничивающих рамок и показатель уверенности (англ. confidence score) для каждой ячейки. Оценка классификации будет варьироваться от 0.0 до 1.0, при этом 0.0 будет самым низким уровнем уверенности, а 1.0 – самым высоким. Если в этой ячейке нет объекта, оценка уверенности должна быть 0.0, а если модель полностью уверена в своем предсказании, оценка должна быть 1.0. Эти значения отражают уверенность модели в том, что в данной ячейке существует объект и что ограничивающая рамка является точной. Каждая из этих границ состоит из 5 чисел: координаты центра границы (x, y), относительные значения ширины и высоты и уверенность.

Значение уверенности представляет собой метрику IOU (Intersection Over Union) между предсказанной границей и фактической границей, предварительно размеченной человеком. Данная метрика представляет собой площадь пересечения предсказанной и истинной границ, деленную на площадь объединения тех же предсказанной и истинной границ.

На рисунке 17 ниже изображена визуализация расчета IOU. Площадь пересечения истинного и предсказанного ящика, выделенная зеленым цветом, разделенная на площадь объединения этих двух ящиков, выделенную фиолетовым цветом. Это значение будет от 0 до 1, 0, если они вообще не пересекаются, и 1, если это один и тот же ящик. Поэтому более высокий IOU лучше, так как это более точное предсказание

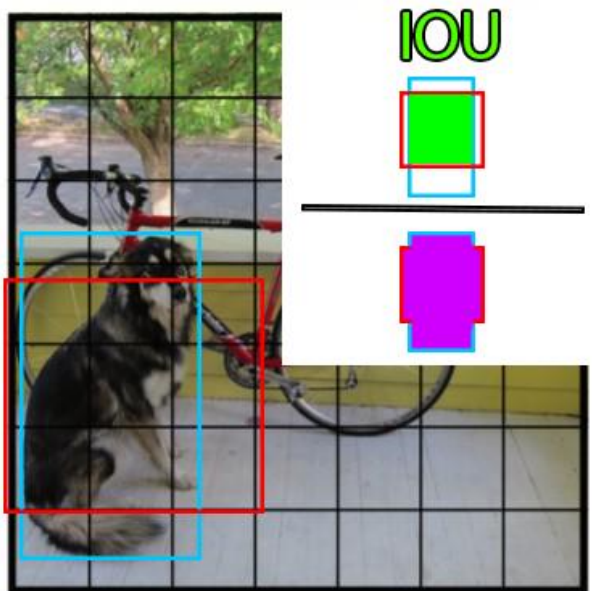

Рисунок 17 – Пример расчета метрики IOU

Кроме вывода значений границы и показателя уверенности для каждой ячейки сеть предсказывает класс, количество которых определяется заранее. Для каждой ячейки может быть предсказан только один класс, это ограничения алгоритма.

На рисунке 18 приведено еще одно изображение всех граничных областей и прогнозов классов и их конечный результат.

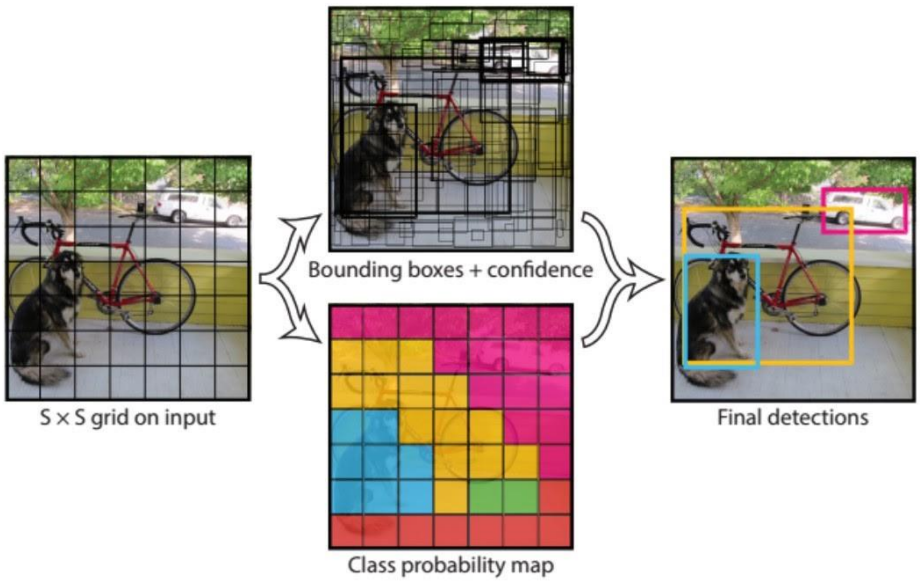

Рисунок 18 – Пример работы сети

Модель YOLO состоит из трех ключевых компонентов: конечных слоев или голов (англ. head), промежуточной части (англ. neck) и экстрактора признаков (англ. backbone). Экстрактор — это часть сети, состоящая из свёрточных слоев для обнаружения ключевых особенностей изображения и их обработки. Костяк сначала обучается на наборе данных для классификации, например, ImageNet, и обычно обучается на более низком разрешении, чем конечная модель обнаружения, поскольку обнаружение требует более тонких деталей, чем классификация. Промежуточная часть использует признаки, полученные от предыдущих слоев для прогнозирования вероятностей и координат ограничивающей области. Голова — это последний выходной слой сети, который может быть заменен другими слоями с той же формой входа для трансферного обучения. Эти три части модели работают вместе, чтобы сначала

извлечь из изображения ключевые визуальные признаки, затем классифицировать и связать их.

Для того, чтобы среди множества предсказанных границ отобразить нужную, применятся механизм Non-maximum Suppression или NMS. Его суть в том, чтобы итеративным способом найти границу с максимальным значением IOU (рисунок 19) [18].

Рисунок 19 – Результат работы механизма NMS

Этот механизм является популярным и используется во многих сетях для обнаружения объектов, поэтому во многих библиотеках в виде функции, что снимает необходимость его отдельной реализации.

2.6.2 Модель для сегментации UNet

Данная архитектура развилась из традиционных свёрточных сетей и изначально использовалась для работы в области медицины. Если при классической классификации изображений ответом являлась только метка (класс) для изображения, то на биомедицинских изображениях, например снимках МРТ, необходимо было не только определить наличие какой-либо патологии, но локализовать место. Это и есть задача сегментации: провести классификацию не по всему изображению, а по каждому пикселю. Таким

образом на выходе будет получатся такое же изображение, как и на входе, но с выделенными пикселями объекта.

Сеть имеет название UNet, потому что её архитектура похожа на английскую букву «U» [14].

В отличие от архитектуры YOLO данная сеть является симметричной и состоит из двух частей: энкодера и декодера. Энкодер – это левая часть сети, которая преобразует исходное изображение в признаки с помощью операции свёртки. Декодер – правая часть, которая восстанавливает изображение с выделенными признаками, используя операцию транспонированной свертки. Эта операция имитирует обратную свёртке операцию с помощью отступа и шага свёртки.

Выход сети – маска изображения со значениями 0 или 1 для каждого пикселя, которые чаще всего конвертируются в черный и белый цвета соответственно (Рисунок 20).

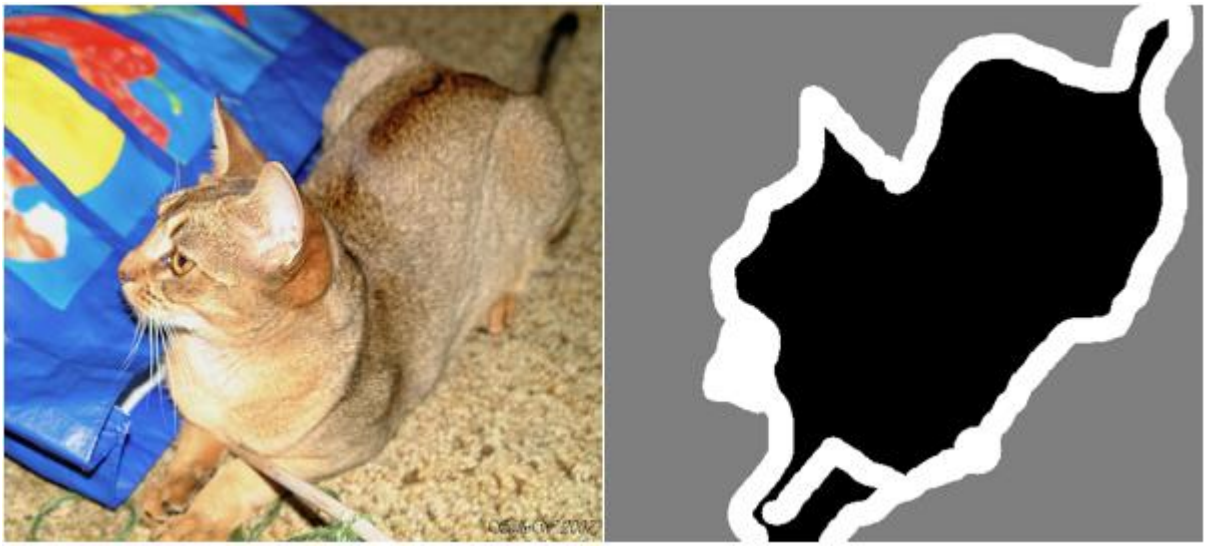

Рисунок 20 – Пример работы UNet.
Серый цвет добавлен для контрастности.

Далее рассмотрим процесс составления наборов данных для рассмотренных выше архитектур, коснемся аугментации данных и проблем обучения.

2.7 Составление наборов данных

Приступим к описанию датасета для задачи.

Датасет будет состоять из изображений людей в очках с разными положениями головы. Составление датасета происходило в несколько этапов. На первом этапе целью был сбор изображений различных людей с разными оправами. На втором решалась проблема переобучения под лицо.

Итак, первыми требованиями к набору данных стали: наличие как можно большего числа различных вариаций оправ очков, так как их разновидностей существует огромное количество; наличие в датасете разных людей, потому что черты лица – очень значимый признак, и сеть с легкостью может переобучиться под конкретное лицо.

Для ускорения и облегчения процесса сбора был написан код на Python, который автоматически разбивал исходное видео на кадры и сохранял их в отдельную папку [17].

Для работы с изображениями была выбрана библиотека OpenCV. Это библиотека с открытым кодом, которая содержит большое количество как методов работы с изображениями и видео, так и алгоритмов, связанных с областью компьютерного зрения [20].

Для записи видео использовалась фронтальная камера телефона, так как предполагалось, что люди будут снимать себя сами. На рисунке 21 приведен пример изображения из датасета.

Рисунок 21 – Пример изображения из набора данных

Кроме описанных выше требований также важно было наличие в данных не совсем обычных положений головы, как на рисунка выше. Это необходимо для того, чтобы сеть правильно находила регион с очками и её результат сети был более стабильным.

Для того, чтобы обеспечить наличие разнообразия оправ, заказчиком были отправлены несколько партий оправ.

Особое внимание уделялось менее заметным и более тонким оправам, так как было предположение, что модель будет хуже определять положение таких типов на видео

Итоговый размер датасета на первом этапе составил 1750 изображений.

Подробнее о выбранных моделях и методе обучения будет рассказано в следующей части. Сейчас остановимся на втором этапе сбора данных.

После первых тестирований модели было обнаружено, что иногда модель определяла, что очки присутствуют, хотя их на самом деле не было (Рисунок 22).

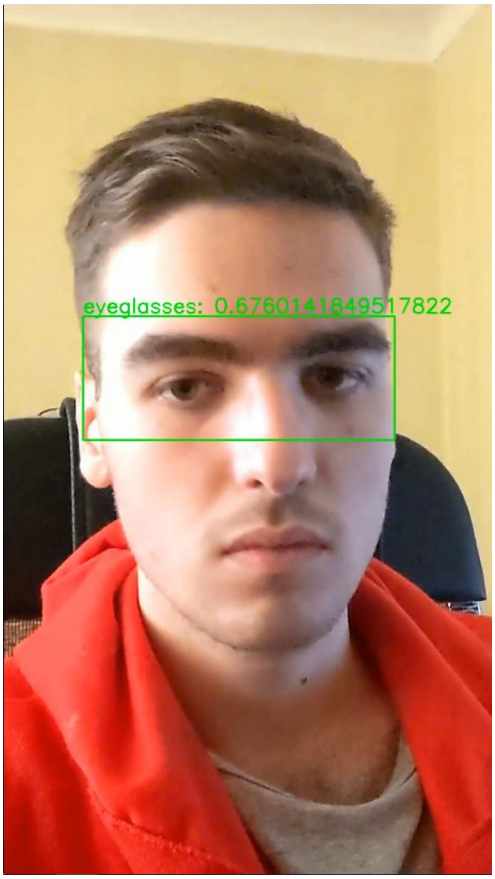

Рисунок 22 – Переобученность модели

Для решения этой проблемы было решено переработать датасет и добавить в него изображения с очками без лиц и с очками не на лице, чтобы модель постаралась запомнить сами признаки очков, удалив при этом часть изображений с очками на лице.

После переработки датасета финальная версия стала содержать 2450 изображений с общим размером 1 гигабайт.

Для разметки данных использовалось приложение с открытым кодом LabelImg. Оно позволяет вручную выделать прямоугольные области н картинке, сохраняя разметку сразу в нужном для YOLO формате в отдельный файл (Рисунок 23).

Рисунок 23 – Пример размеченной картинки и файла с разметкой

Для сбора данных для второй сети использовалось специальное приложение заказчика, которое позволяет виртуальные очки. Оно добавляет на изображение одну из заранее добавленных моделей очков. Таким образом были получены пары изображений с очками и без (Рисунок 24).

Рисунок 24 – Пример данных для сети UNet.

Далее с помощью библиотеки OpenCV высчитывалась абсолютная разница между изображениями, проводилась бинаризация полученной маски (преобразование всех пикселей в белый или черный по пороговому значению). Пример итоговой маски для оправы выше на рисунке 25.

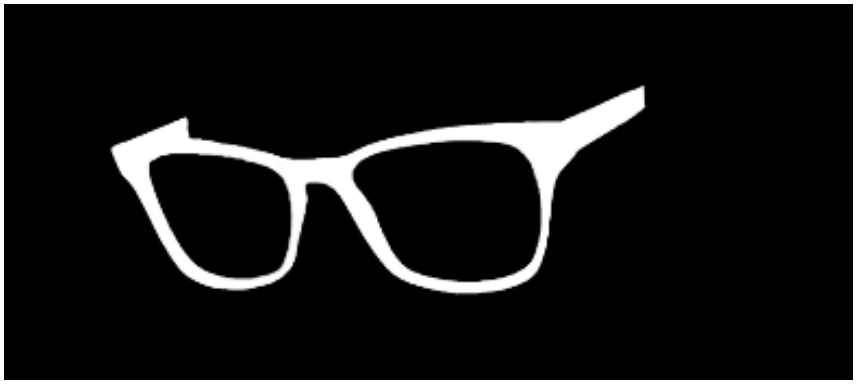

Рисунок 25 – Бинарная сегментационная маска для оправы

При таком подходе цветные и неконтрастные оправы давали рваные маски с шумом вокруг оправы (Рисунок 26). Такие изображения дорабатывались вручную.

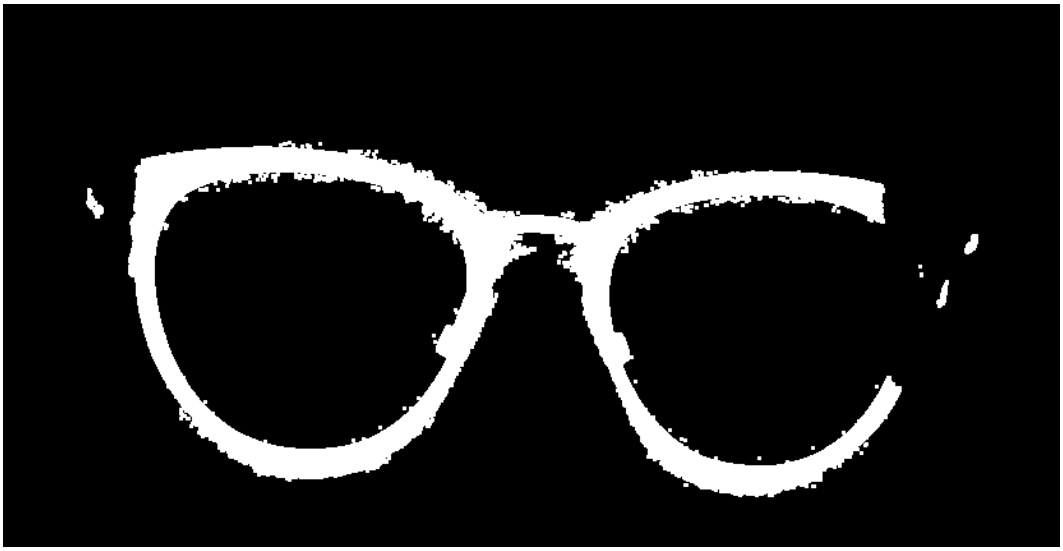

Рисунок 26 – Рваная маска с шумом

Размер набора данных для сегментации составил порядка 1.5 тысяч изображений.

2.7.1 Аугментация данных, проблемы обучения моделей

В случае, если нет возможности собрать достаточное количество данных самим, поэтому здесь приходит на помощь аугментация данных. Аугментация – это способы изменения исходного изображения для создания достаточного количества данных для тренировки моделей. Наиболее распространёнными вариантами изменения являются изменения параметров цвета, насыщенности, контрастности, гаммы; добавление размытия, отображения по вертикали/горизонтали, масштабирование. На рисунке 27 ниже изображены данные преобразования

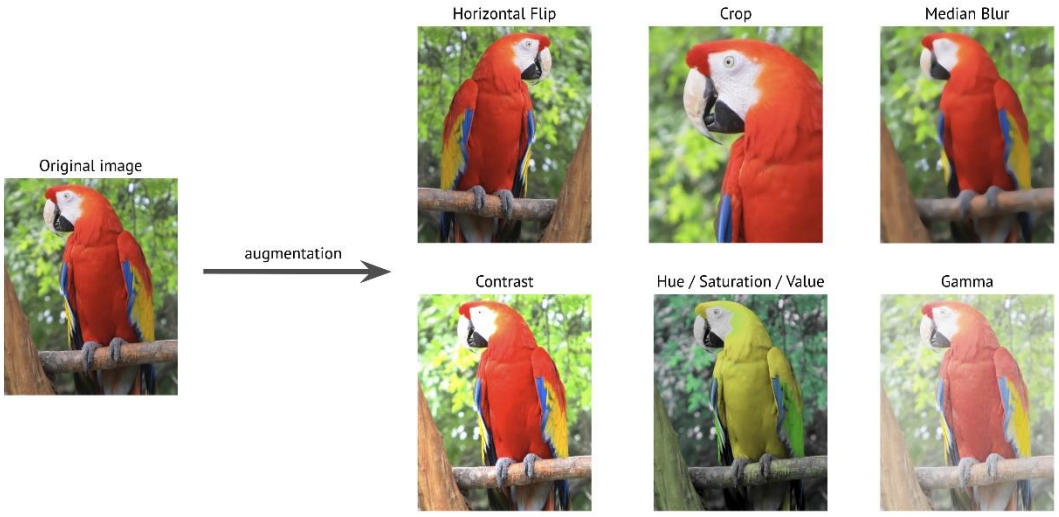

Рисунок 27 – Пример аугментации изображения

В моём случае аугментация встроена во фреймворк для обучения, о котором будет рассказано позже.

Но что, если данных для обучения недостаточно? Сможет ли модель обобщить данные и выделить закономерности? Инструменты для обучения сами по себе не запрещают обучать модели на малом количестве данных. Здесь возникают проблемы переобучения и недообучения. На рисунке 28 схематично изображены результаты хорошего обучения и двух его крайностей [2].

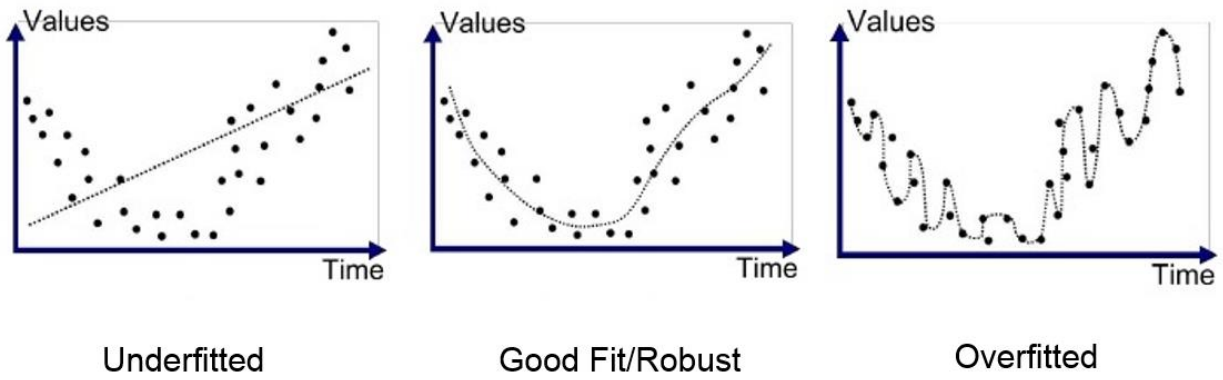

Рисунок 28 – Недообучение, оптимальный результат и переобучение

Переобучение – ситуация, при которой модель слишком сильно запомнила признаки тренировочной выборки. При переобучении точность модели на такой выборке может достигнуть 100% в худших случаях. Но при проверке на новой, ранее не использовавшейся части данных, результат окажется печальным. Модель не смогла выделить общие закономерности в данных, запомнив конкретные примеры и ожидая таких же в будущем.

Обратная ситуация – это недообучение. В этом случае модель не выделила признаки из данных, и её точность будет низкой как на тренировочной выборке. Так и на новых данных.

Эти проблемы чаще всего можно решить увеличением размера датасета, очисткой данных от шума и нерелевантных примеров или созданием новых признаков. Кроме этого, если манипуляции с данными не помогли и есть такая возможность, можно заменить модель на более простую (для переобучения), а также уменьшить регуляризацию модели или заменить ее на более сложную (для недообучения).

2.8 Фреймворки для работы с нейронными сетями, обучение моделей

Перейдем непосредственно к инструментам и процессу обучения.

Для обучения модели YOLO использовался фреймфорк под названием darknet. Данный фреймворк разработан тем же человеком, который создал четвёртую версию алгоритма. Данный фреймворк использует подход транзитного обучения.

Само процесс осуществляется посредством ввода команд в консоль операционной системы. Во время обучения так же выводится график с основными показателями: средняя точность по всем классам, значение ошибки, номер итерации и оставшееся время.

Прежде чем запустить команды, необходимо предварительно иметь файл конфигурации и предварительно натренированные веса. На моем

компьютере обучение занимало около 4 часов. Результатом обучения являются файлы весов с расширением. Эти веса, вместе с файлами конфигурации будут использоваться в дальнейшем при обработке видео.

Для обучения модели UNet использовалась библиотека с открытым исходным Keras. Она используется как интерфейс для другой популярной библиотеки, разрабатываемой сообществом Google, под названием TensorFlow. Для подготовки к обучению необходимо написать код на языке Python. Само обучение, в отличие от предыдущего фреймворка, производится с помощью запуска встроенного метода «fit». Кроме того, архитектура сети так же описывается в коде [15], [19].

По завершении обучения, которое в среднем длилось 3 часа, создается файл модели, в котором хранятся и веса, и конфигурация. Этот файл также будет использоваться в дальнейшем.

2.9 Использование обученных моделей в коде

Комплекс представляет собой консольное приложение из трех методов, где пользователю необходимо необходимые пути и имена файлов (рисунок 29).

Видео обрабатывается следующим образом:

- каждый кадр видео сжимается до заранее установленного разрешения 416x416 и подаётся на вход сети для обнаружения очков. Результатом являются массивы с координатами регионов, метками класса и значениями уверенности;
- все ненужные координаты и соответствующие им значения класса и уверенности отфильтровываются с помощью функции, реализующей механизм NMS В результате, получаются наиболее верные координаты региона с очками;

– после этого регион с очками вырезается из кадра, изменяется до нужного разрешения и подаётся на вход сети для сегментации. В результате получается бинарная маска очков;
– далее маска увеличивается до размера региона с очками, и накладывается на этот регион;
– регион с наложенной маской подаётся на вход функции восстановления изображения;
– результат восстановления накладывается на оригинальный кадр, который записывается в выходное видео;
– программа работает, пока не закончатся кадры исходного видео.

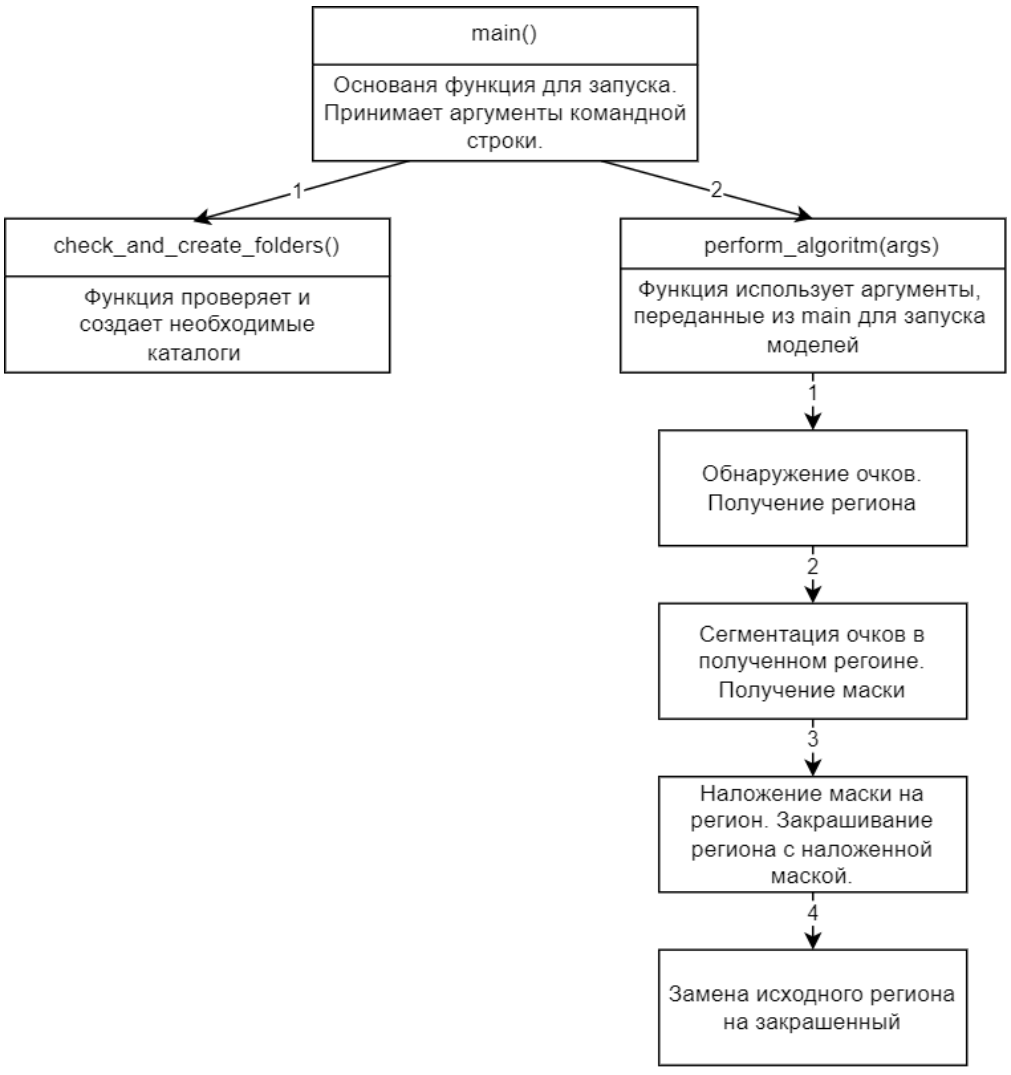

Рисунок 29 – Структура программы

Для работы модели обнаружения и для обработки входных файлов используется библиотека OpenCV, которая предварительно скомпилирована для поддержки видеокарты. В ней присутствует встроенный модуль для работы с большим количеством нейронных сетей. Для второй модели использовалась библиотека Keras.

Выводы и результаты по главе 2

В этой главе были определены этапы жизненного цикла и разработки программы, в соответствии с которыми были выбраны и рассмотрены модели нейронных сетей, описан процесс сбора данных и его разметка, были описаны инструменты для обучения моделей и процесс обучения, описана структура и принцип работы комплекса. Также дано описание предметной области и теоретические сведения, необходимы для разработки.

Глава 3. Тестирование программного комплекса

3.1 Проверка работы комплекса на изображениях и видео

Тестирование проводилось видео, снятых на фронтальную камеру телефона. Все видео для тестирования не включались в наборы данных для обучения модели. Тестирование производилось на видео со мной и с членами команды, работавшей над другими частями проекта.

Время работы программы на видео с разрешением 1920x1080 составляет примерно 2.5 минуты. Частота кадров и соотношение сторон после обработки нейросетью не изменяется.

На рисунке 30 представлены результаты для темной контрастной оправы. На таких типах оправ комплекс работает лучше всего, потому что их достаточно просто находить из-за явно разницы с оттенком лица. В районе очков наблюдается еле заметное «размывание» изображения. Связано это с тем, что регион для работы подается в низком разрешении, чтобы нейросети смогли его обработать, после чего он расширяется до изначального соотношения сторон. Это расширение делается с помощью заполнения пикселей средним значением вокруг него, поэтому белая маска оправы становится чуть толще и некоторые детали пропадают.

На рисунке 31 изображен результат работы на светлой оправе. Такие оправы являются более сложным случаем. Тем не менее, алгоритм справляется с поставленной задачей.

Рисунок 30 – Темная оправа

Рисунок 31 – Светлая оправа

На рисунке 32 – результат теста в маске. Программа успешно справилась с задачей и закрасила очки. Верхняя часть маски слегка изменилась, но это не критично, так как после наложения виртуальных очков эта область скроется под ними. Так же сами очки являются самым сложным случаем. Потому что нижняя часть оправы, в отличие от предыдущих случаев, заменена на леску, и, по сути, отсутствует. Алгоритм стал работать на таких типах оправ после составления датасета практически полностью состоящего из изображений с похожими тонкими оправами.

Рисунок 32 – Тонкая оправа и маска

Как видно на рисунке 33, сами линзы очков и, соответственно, отражения в них, не закрашиваются. Сделано это намеренно, так как находить линзы – задача практически невыполнимая, потому что при фронтальном взгляде в камеру в линзы не отражают ничего, и определить их наличие очень

трудно даже для человека. Плюс к этому, линзы перекрывают довольно большую часть лица, и их закрашивание превратило бы глаза в непонятную кашу.

Рисунок 33 – Отражения в линзах очков

По итогам тестирования, подход с использованием комбинации нейронных сетей и алгоритма компьютерного зрения оказался успешным. Комплекс показывает неплохие результаты на разных типах оправ, не оказывая существенного влияния на качество изображения.

3.2 Возможные варианты улучшения и место применения комплекса

Основным улучшением, помимо увеличения набора данных, является замена классического алгоритма закрашивания на нейронную сеть. Самым

подходящим кандидатом для такой замены является Генеративно-состязательная сеть (англ. Generative adversarial network, сокращённо GAN). Данный тип сети состоит из двух сетей, одна из которых генерирует данные, а другая – пытается отличить их от реальных данных. Это позволяет в конечном итоге создавать данные из того же распределения, что и оригинальные. Данная архитектура используется для создания реалистичных фотографий. Существует модификация данной сети под название «AOT-GAN», которая используется для закрашивания изображений. Кроме этого, можно заменить сеть для обнаружения на ее «старшую» модель без индекса “tiny”. Она имеет большее количество свёрточных слоев и дает более точные результаты. Но такая замена может существенно повлиять на производительность.

Данный программный комплекс будет использоваться в мобильных телефонах компании Apple. Связано это с тем, что в новых моделях стал устанавливаться отдельный нейронный процессор именно для работы с нейросетями. Данный процессор позволяет существенно ускорить работу сетей и снижает нагрузку на графическую карту и основной процессор. В системе Android реализация будет сложнее несмотря на то, что существуют мобильные версии библиотек для обучения. В них нет отдельного нейропроцесора, и для каждой марки телефона алгоритм реализации свой, что делает перенос весьма трудоемким процессом.

Выводы и результаты по главе 3

В данной главе были представлены результаты работы комплекса на различных оправах и предложен вариант улучшения комплекса с помощью дополнительной нейронной сети.

Заключение

В результате выполнения данной работы была достигнута цель и создан программный комплекс для стирания очков с лица, основными компонентами которого являются нейронные сети и классический алгоритм компьютерного зрения. Был проведен анализ компании-заказчика, определены требования к комплексу, сформулированы этапы жизненного цикла и разработки.

Были решены следующие задачи:

- проанализирован основной бизнес-процесс компании, для улучшения которого был разработано программный комплекс, смоделирована диаграмма «как есть»;
- сформулирована задача на разработку;
- разработана и проанализирована диаграмма «как должно быть»;
- выбрана архитектура ИНС и основанные на ней модели;
- с помощью нейронных сетей решены задачи нахождения и сегментации объекта на изображении;
- собраны необходимые данные;
- обучены несколько вариантов моделей;
- подобран классический алгоритм для закрашивания объекта
- написан программный код для тестирования комплекса.

В ходе работы с данными было определено, что этап сборки, предобработки и разметки данных при использовании искусственных нейронных сетей оказывает существенное влияние на конечный результат.

Данный комплекс довольно хорошо решает поставленную задачу, несмотря на несовершенство классического алгоритма и то, что нейронные сети, хоть и обладая большой обобщающей способностью, не дают 100% точных результатов.

Список используемой литературы и используемых источников

1. Компьютерное зрение [Электронный ресурс]: URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B7%D1%80%D0%B5%D0%BD%D0%B8%D0%B5 (дата обращения: 25.04.2022).

2. Переобучение [Электронный ресурс]: URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5#:~:text=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20(%D0%BF%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%B4%D0%B3%D0%BE%D0%BD%D0%BA%D0%B0%2C%20%D0%BF%D0%B5%D1%80%D0%B5%2D%20%D0%B2,%D0%BD%D0%B0%20%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%B0%D1%85%20%D0%B8%D0%B7%20%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B9%20%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8) (дата обращения: 25.04.2022).

3. Свёрточная нейронная сеть [Электронный ресурс]: URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%91%D1%80%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C (дата обращения: 25.05.2022).

4. Сверточные нейронные сети [Электронный ресурс]: URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D0%B2%D0%B5%D1%80%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8 (дата обращения: 15.04.2022).

5. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество [Электронный ресурс]: URL: https://habr.com/ru/post/348000/ (дата обращения: 02.06.2022).

6. Ananda Hange. Target Prediction using Single-layer Perceptron and Multilayer Perceptron [Электронный ресурс]: URL: https://medium.com/nerd-for-tech/flux-prediction-using-single-layer-perceptron-and-multilayer-perceptron-cf82c1341c33 (дата обращения: 13.03.2022).

7. Ani Aggarwal. YOLO Explained [Электронный ресурс]: URL: https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31 (дата обращения: 02.04.2022).

8. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, – US: O'Reilly Media, 2019 – 510

9. CNN | Introduction to Pooling Layer [Электронный ресурс]: URL: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/#:~:text=Max%20pooling%20is%20a%20pooling,of%20the%20previous%20feature%20map (дата обращения: 25.04.2022).

10. Convolutional neural network [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/Convolutional_neural_network#Convolutional_layer (дата обращения: 10.03.2022).

11. Harshall Lamba. Understanding Semantic Segmentation with UNET [Электронный ресурс]: URL: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47 (дата обращения: 25.04.2022).

12. ImageNet [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/ImageNet (дата обращения: 15.04.2022).

13. Inpainting [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/Inpainting (дата обращения: 06.04.2022).

14. Jeremy Zhang. UNet — Line by Line Explanation [Электронный ресурс]: URL: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5 (дата обращения: 25.04.2022).

15. Keras [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/Keras

16. Pulkit Sharma. Image Classification vs. Object Detection vs. Image Segmentation [Электронный ресурс]: URL: https://medium.com/analytics-

vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81 (дата обращения: 25.04.2022).

17. Python 3.10.4 documentation [Электронный ресурс]: URL: https://docs.python.org/3.10/ (дата обращения: 13.05.2022).

18. Sambasivarao. K. Non-maximum Suppression (NMS) [Электронный ресурс]: URL: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c (дата обращения: 25.04.2022).

19. TensorFlow [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/TensorFlow (дата обращения: 25.04.2022).

20. OpenCV [Электронный ресурс]: URL: https://docs.opencv.org/4.5.5/d1/dfb/intro.html (дата обращения: 25.04.2022).