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)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Применение сверточных нейронных сетей для локализации объектов на изображении»

Студент Е.А. Гвоздецкий
(И.О. Фамилия) (личная подпись)

Научный руководитель доцент, В.С. Климов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Содержание

Введение

Глубокое обучение – это подмножество методов машинного обучения, области изучения и создания программных продуктов, алгоритмов, которые могут обучаться. Иногда с целью достичь уровня искусственного интеллекта.

Машинное обучение – это область компьютерных наук, в которой машины учатся решать задачи, для которых они не были запрограммированы непосредственно сами.

Глубокое обучение используется в промышленности для решения практических задач в самых разных областях, таких как компьютерное зрение (изображение, видео), обработка естественного языка (текст) и автоматическое распознавание речи. Глубокое обучение главным образом основано на применении искусственных нейронных сетей, которые представляют класс алгоритмов, имитирующих работу человеческих нейронных сетей. Главным преимуществом глубокого обучения над другими методами машинного обучения является высокая степень автоматизации нахождения признаков предметной области, то есть результат работы алгоритмов нейронных сети будет зависит главным образом от количества и качества данных на подаваемых на этапе обучения алгоритма. Это делает алгоритмы глубокого обучения универсальными, их можно обучить решать задачи на одном массиве данных, а также используя другой массив данных алгоритм сможет решать другого радо задачи. Машинное обучение является активно развивающейся областью и нейронные сети, которые входят в область машинного обучения являются актуальными для исследования.

Сверочные нейронные сети на сегодняшний момент являются лучшим способом автоматизации задач, которые требует участия человека, либо проектированием и разработкой алгоритмов сложный программных модулей, которые требует больших затрат.

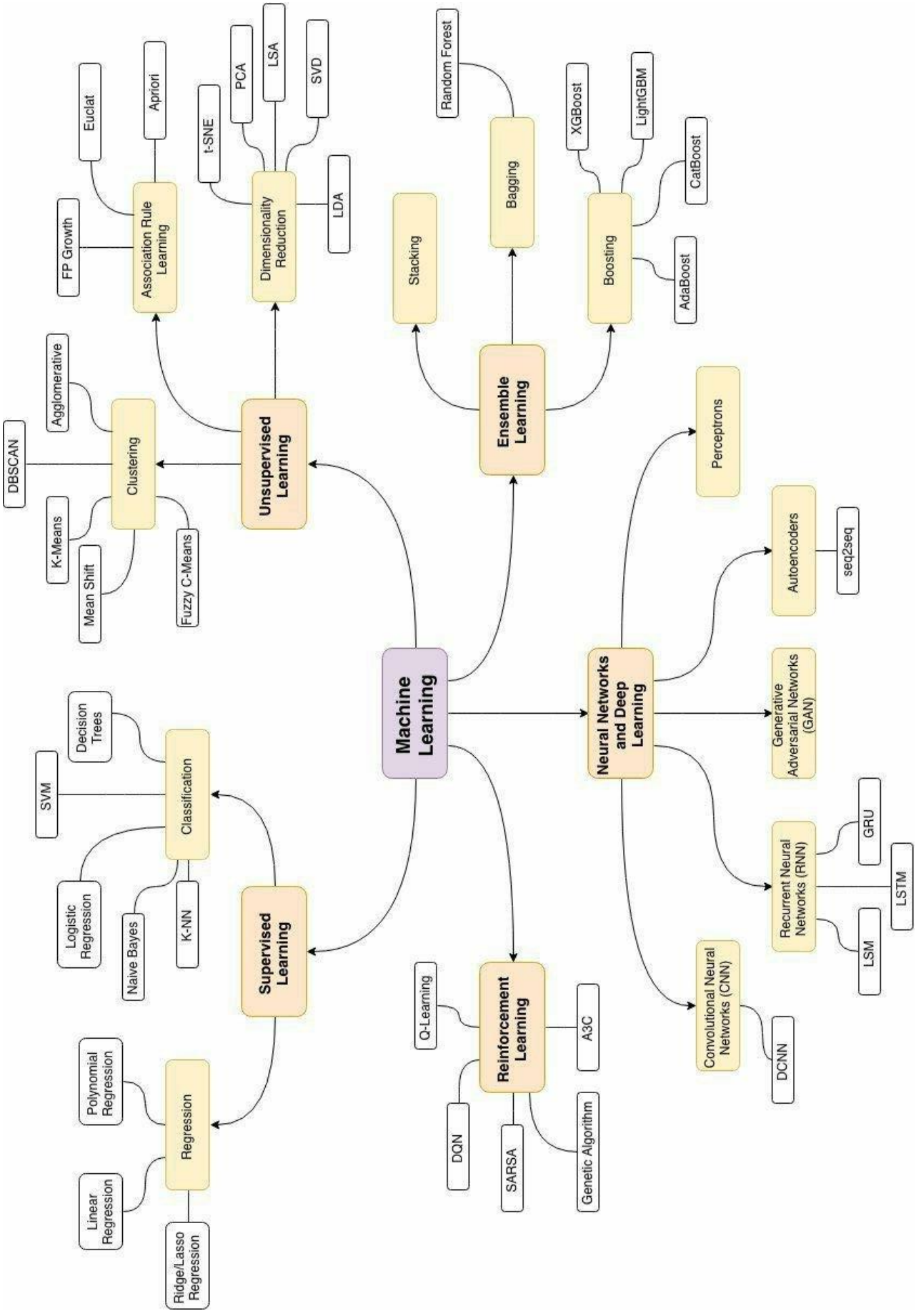

Рисунок 1.1 – Виды алгоритмов машинного обучения

В области машинного обучения существует отдельное семейство методов глубокого обучения, разделенный на несколько алгоритмов. Сверточнные нейронные сети проектировались изначально, как сети для классификации изображений, но потом было выяснено, что они могет справляться и с другими задачами Машиного обучения, начиная от локализации объекта, заканчивая предсказанием последовательности текста. Сверточнные нейронные сети получили свою популярность, благодаря того, что они способны рассматривать совокупность признаков на изображении, в отличии от персептрона, где, каждый нейрон пытается рассматривать свой пиксель, и из-за этого теряется пространство изображения. Все виды нейронный сетей представлено на рисунке 1.2.

Гипотезой исследования является предположение, что используя методы свёрточнных нейронных сетей, можно научить алгоритм автоматизировать задачу локализации объектов на изображении, без разработки сложных алгоритмов.

Объектом исследования являются локализация объектов на изображении, предметом исследования – разработка алгоритма для локализации объектов на изображении с использованием свёрточнных нейронных сетей.

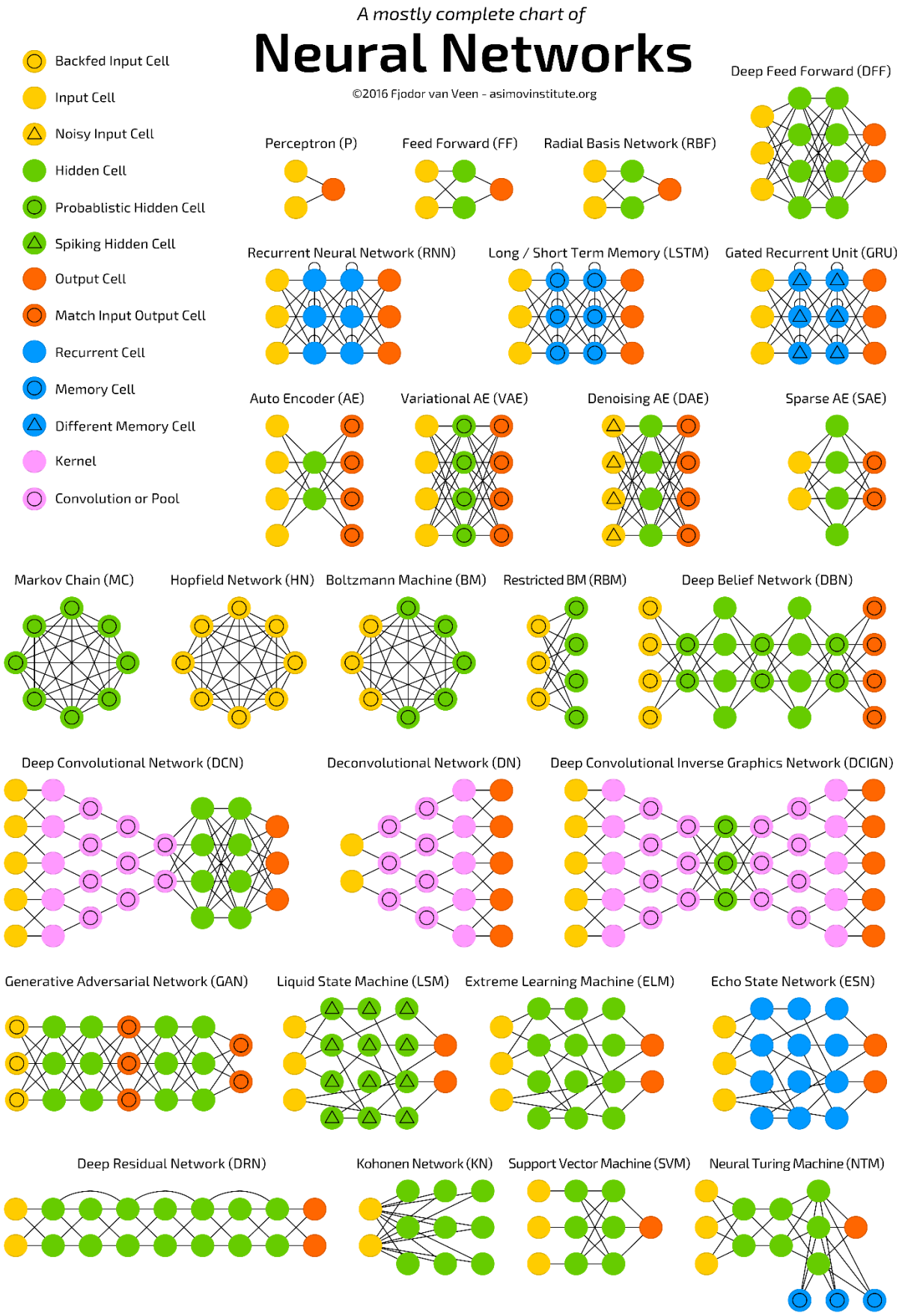

Рисунок 1.2 – Виды алгоритмов машинного обучения

Цель исследования – использование алгоритма сверточнных нейронных сетей, для автоматизации задачи по локализации объектов на изображении.

Цель исследования выполняется решением задач, перечисленных ниже:

1. Анализ вопроса темы магистерской дисертации.

2. Разработка математической модели для локализации объектов на изображении (при анализе данных с помощью алгоритма свёрточнных нейронных сетей).

3. Проектирование и разработка математической модели данного метода для оценки эффективности задачи на практике.

4. Тестирование математической модели на практике. Получения выводов результата практических экспериментов.

Научная новизна исследования – доказано, что в машинном зрении точность локализации объектов на изображении и в реальном времени может быть увеличена с применением сверточнных нейронных сетей и увеличена скорость работы алгоритма с использованием графических процессоров для математических вычислений в работе алгоритма.

Практическая значимость работы заключается в разработке математической модели сверточнной нейронной сети для локализации объектов, что должно увеличить точность локализации машинного зрения и ускорения работы за счет использования графических процессоров в работе математической модели.

В первой главе исследования рассматриваются проблемы развития алгоритмов машинного обучения. Проблемы задачи машинного зрения. Проблемы работы нейронных сетей и глубокого обучения.

Во второй главе рассматривается формальное описание задачи моделирования сверточнной нейронной сети. Представлена математическая модель. Проблемы с переобучением, которые возникают при обучении на малых выборках и способы их решения. Выбор нейронной сети, которая способна решать максимально эффективно задачу локализации объектов и

выполнения требований под реализацию. Результаты экспериментов при реализации математической модели нейронной сети.

В третьей главе представлены результаты программного обеспечения. Спроектирована реализация математической модели локализации объектов на изображении и практическая польза, использую как модуль в задаче распознавания людей по лицу.

На защиту выносятся:

1. Математическая модель сверточнной нейронной сети, которая способна решать задачу локализации объектов.
2. Результаты практических реализаций научного исследования и программного обеспечения и практическая значимость в виде программы по идентификации людей по лицу.

По результатам проведенных исследований опубликовано 5 статей в сборниках:

Результаты исследований доложены на таких конференциях как:

- Всероссийская научно-практическая междисциплинарная конференция «Молодежь. Наука. Общество», г. Тольятти.
- V Международной научно-практической конференции (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук», г. Тольятти
- II выставки-конкурсе разработок СКБ/СНИЛ опорных вузов, г. Омск

1 АНАЛИЗ СОСТОЯНИЯ ВОПРОСА

1.1 ПРОБЛЕМЫ РАЗВИТИЯ АЛГОРИТМОВ ГЛУБОКОГО ОБУЧЕНИЯ

Машинное обучение - это раздел информационных технологий, который частично охватывает раздел искусственного интеллекта. При этом машинное обучение не является классическим примером искусственного интеллекта. Все методы машинного обучения, включая глубокое обучение, представляют из себя алгоритмы, которые способны самостоятельно на этапе обучения выявлять признаки из входных данных. Сам способ обучения делится на два вида: обучение с учителем (с подкреплением), обучение без учителя. Цель любого алгоритма машинного обучения дать предсказание по входным данным.

Машинное обучение имеет огромное семейство, представленное на рисунке 1.1.

Нейронная сеть - попытка воспроизвести работы человеческого мозга на компьютере с использованием связи нейронов. Представляет собой математическую модель, эмитирующую биологическую нейронную сеть.

Глубокое обучение - область использующая нейронные сети, где отличительной способностью является наличие скрытых слоев в количестве более 1. Скрытые слои представляют из себя математические вычисления , используя входные данные. При проектирование нейронных сетей – основной задачей является определение количества скрытых слоев и их размерности нейронов на последующем слое. Определение «глубина» исходит из «глубокое обучение», что утверждает наличие скрытых слоев в размере более 1 слоя.

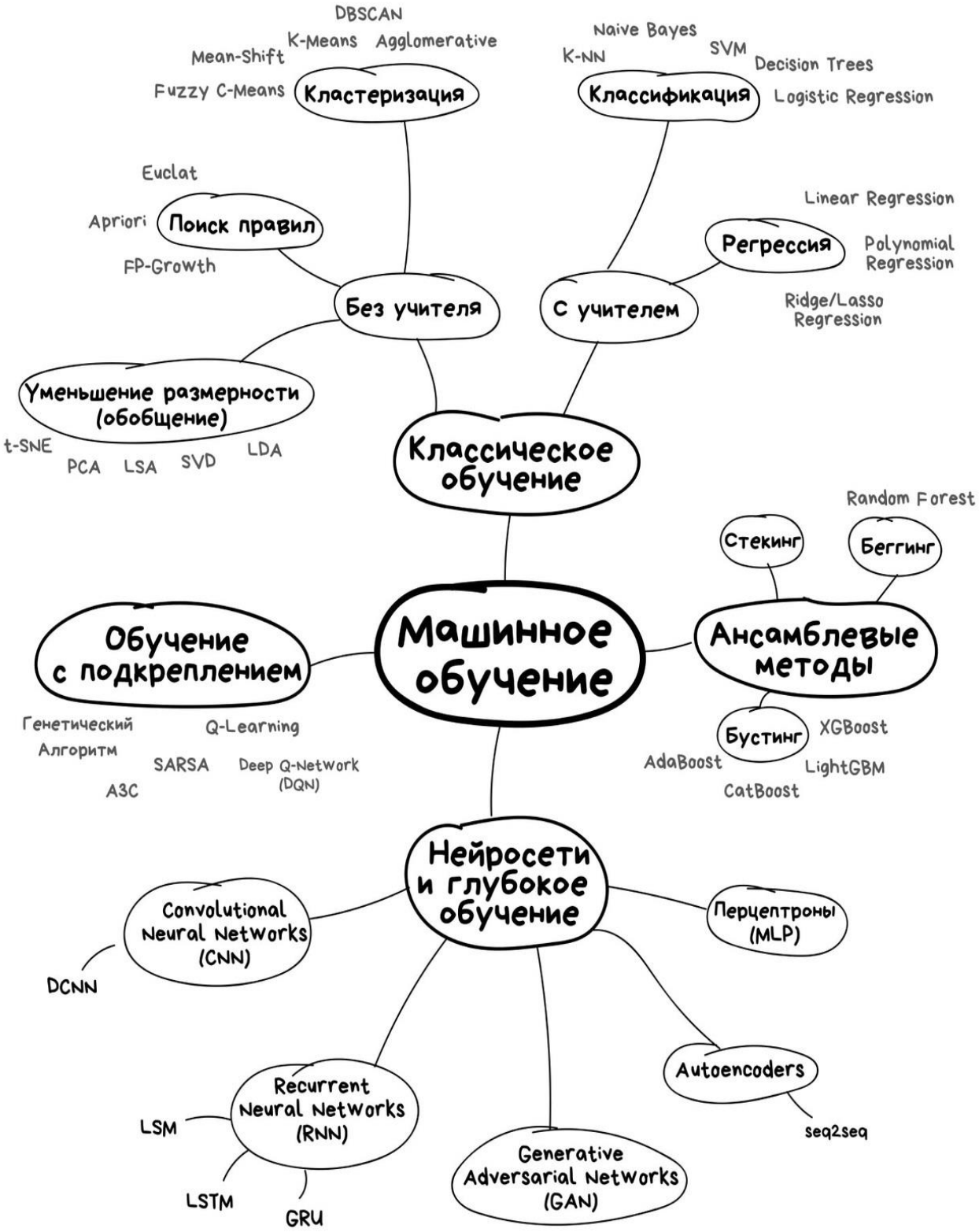

Рисунок 1.1 – Классификация алгоритмов «машинного обучения»

Определение «машинное обучение» (Machine Learning) имеет множество обретений. Наиболее популярное определение говорит, что «Машинное обучение» это набор методов, которые могут решать задачи не напрямую, а находить решение поставленной задачи при обучении используя набор данных реального мира. При проектировании методов «машинного обучения» применяются модели «математической статистики», математические оптимизации и область вероятности, а также численные методы методы.

На сегодняшний день идут активные исследования в области «глубокое обучение» (Deep Learning). Именно к этому классу относятся сверточнные нейронные сети, которые отлично справляются с задачами классификации и локализации изображений. Чтоб понять принцип работы глубокого обучения, рассмотрим архитектуру нейронной сети персептрона, полно связную нейронную сеть.

Полно связной нейронной сети, которая имеет архитектуру при проектировании, где нейрон связан со всеми нейронами предыдущего слоя, каждая связь имеет свой персональный весовой коэффициент. Формула 1.1 вычислений выходного сигнала y нейрона показана ниже, где F – эта функция активации нейрона, Y – выходное значение нейрона, X – входной сигнал (либо пиксель изображения, либо выходное значение другого нейрона и w – значение веса сигнала.

$Y = F(\sum_{i=1}^{n} X_i * w_i)$ – формула 1.1 вычисления выходного сигнала нейрона.

Все параметры функции кроме веса w, являются статичными и задаются при проектирование нейронной сети, либо при данных, которые поступают на вход. Сам параметр w, является настраевым, нейронная сеть в процессе обучения должна подобрать такое значение этого веса, чтоб в дальнейшем при данных на обучении и на новых данных получалось вероятность нейронных сети удовлетворяющих его работой.

Сверточнная нейронная сеть основана на таких же принципах, но расширяет сам нейрон, дополняя хранящеюся информацию в нем.

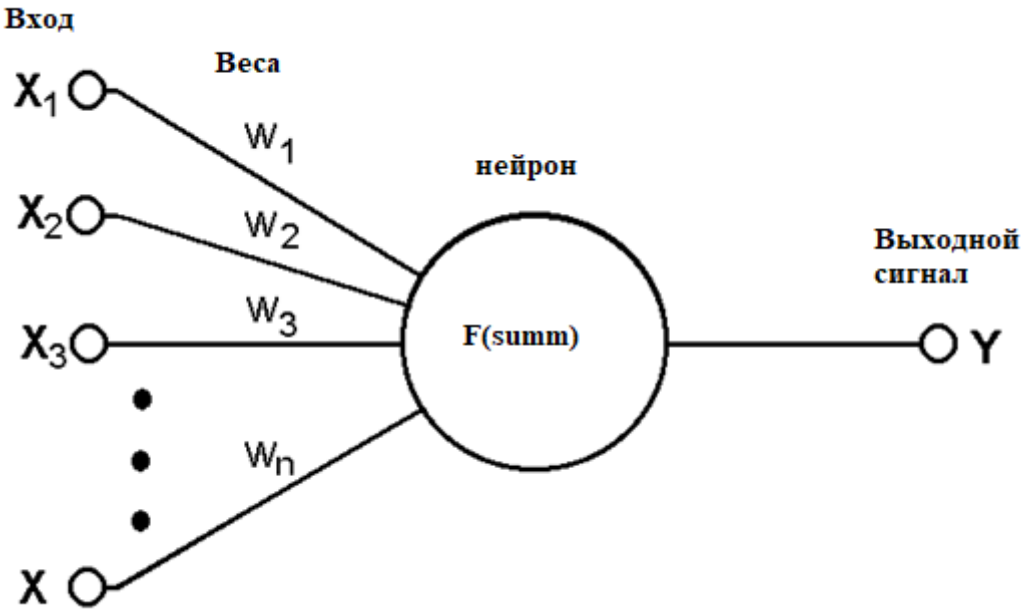

Рисунок 1.2 – Нейрон сверточнной нейронной сети

Выше на рисунке 1.2 представлен схематичное отображение нейрона сети Прецептрон, где изображено подача сигнала на вход, веса у связей, которые могут отличатся от друг друга по значению, тем самым увеличивая, либо уменьшая значимость сигала этого нейрона (Это представляет собой механизм искусственного интеллекта, где программа по набору признаках пытается определять решение), потом сумма всех произведений идет через нелинейную функцию активации, для того чтоб нейрон мог делать предположение почти, возможно, наверняка. В первых версий нейронов использовалась жесткая пороговая функция активации, умеющая делать предположение либо да, либо нет. Более детальный разбор функций активаций представлен ниже.

Нейрон сверточной нейронной сети представляет собой ограниченную квадратичную матрицу весов небольшого размера от 3 до 7 ячеек, эта матрица называется ядром свертки. На вход такого нейрона подается матрица входного изображения, ядро свертки проходит по всей этой

матрицы, формирую после каждого сдвига сигнал активации для нейрона следующего слоя либо с аналогичной позиции, либо со сдвигом, не учитывая края входной матрицы. Благодаря такой процедуре называемой сверткой мы получаем кодирование какого-либо признака, таким образом следующий слой выделает признак, который нашел в предыдущем слое и координаты признака, представляющею матрицу «Карты признаков». «Карты признака», имеющею различные ядра свёртки, выделяя свои признаки на изображении (пример: круги и линии и фигуры), ручная простановка «ядер свертки» практически невозможно, по этой причине разработчик не может их реализовать, а заполняются они методом обратного распространения ошибки при обучении. На рисунке 1.3 изображен визуально нейрон сверточной нейронной сети.

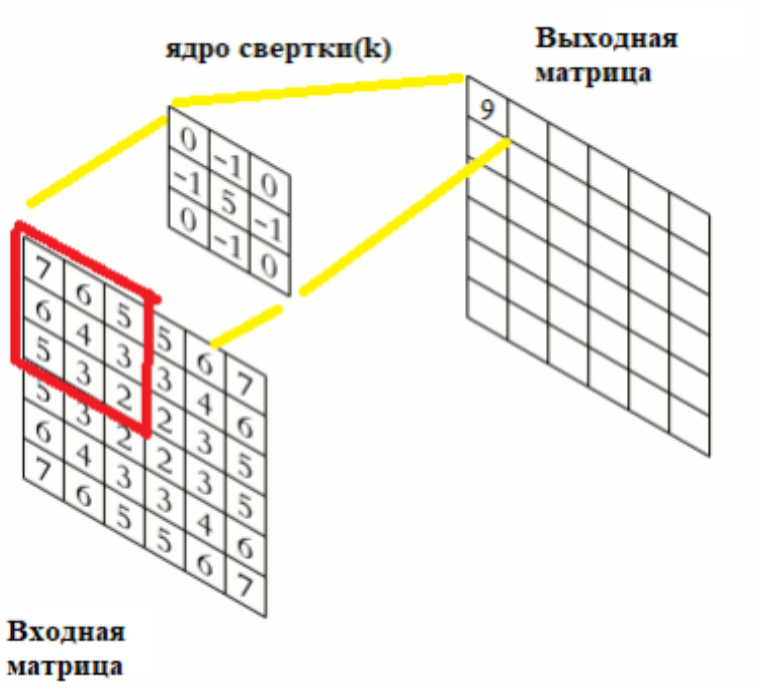

Рисунок 1.3 – Нейрон сверточнной нейронной сети

Алгоритм работы нейрона свертки. Сперва выбирается участок входного изображения слева вверху равного размеру ядра свертки. Происходит умножение каждого значения выделенного фрагмента на каждое значения ядра свертки по их индексам матрицы. В конце все значения

суммируются и выходное значения записывается в ячейку выходной матрицы. Формула приведена ниже.

$$Y = \sum_{i=1,j=1}^{H,W} (X_{i,j} * k_{i,j})$$

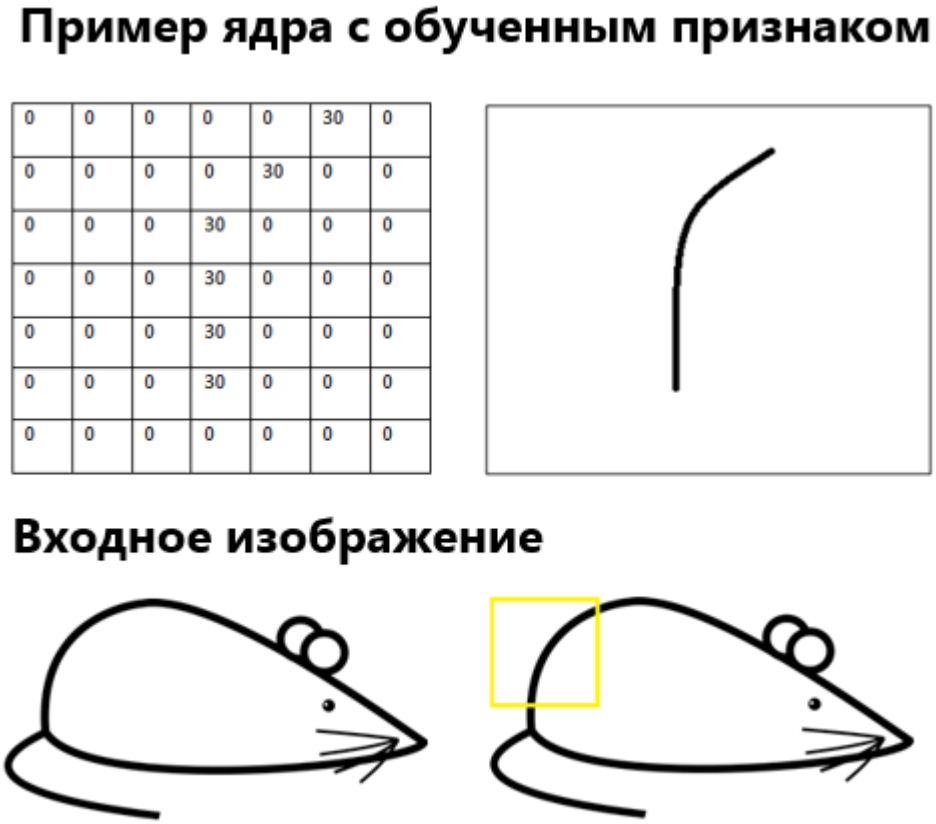

Рисунок 1.4 – Пример результата работы сверточной нейронный сети

На рисунке 1.4 показано, что был получен отклик, в виде линии, что говорит о наличии признака на представленном изображении нейронной сети. При этом на рисунке 1.4 представлена визуализация наглядная работы нейрона, в работе сверточнных нейронных сетей алгоритм работает сравнивания менее абстрактные области.

Рассмотрим преимущества сверточной нейронной сети, сравнивать их будем с полно связной нейронной сетью. Сверточные сети имеют сильный иммунитет к преобразованием объекта на изображении, смене ракурса и другим «искажениям». Они могут хранить информацию о положении найденного признака на изображении. Также сверточнные нейронные сети

хранят более абстрактные данные о поисковом классе объектов, что должно показать более точные результаты, чем применение обычного перцептрона.

Сверточные сети объединяют представляют архитектурные идеи , которые должны обеспечить инвариантность изменения положений и масштабируемости искомого объекта в задача, ниже приведены 3 идеи:

1. Искать локальный признак объекта на изображении для дальнейшего его опрокидывания на следующие слои, если признак слишком мал, слой его игнорирует;
2. Уменьшение параметров нейронной сети за счет «карт признака» по сравнению с классической полно связнной нейронной сетью, где каждый ней рой является параметром сети;
3. Представление каждого слоя более общей математической моделью, имитирующие биологическое зрение.

Структура «сверточной нейронной сети». Состоит слоев которые представлены далее: «сверточные слои», были описаны в магистерской работе выше, слои «под выборки» и слои «перцептрона» полно свзяной нейронной сети которые тоже были представлены выше, на рисунке 4 визуальна изображена структура сверточной нейронной сети вида «Сеть в сети». Количество слоев свертки и подвыборки, обычно равное. Количество слоев полносвязного слоя обычно равно 2-3, но может быть всего 1, не считая слоя преобразования в вектор.

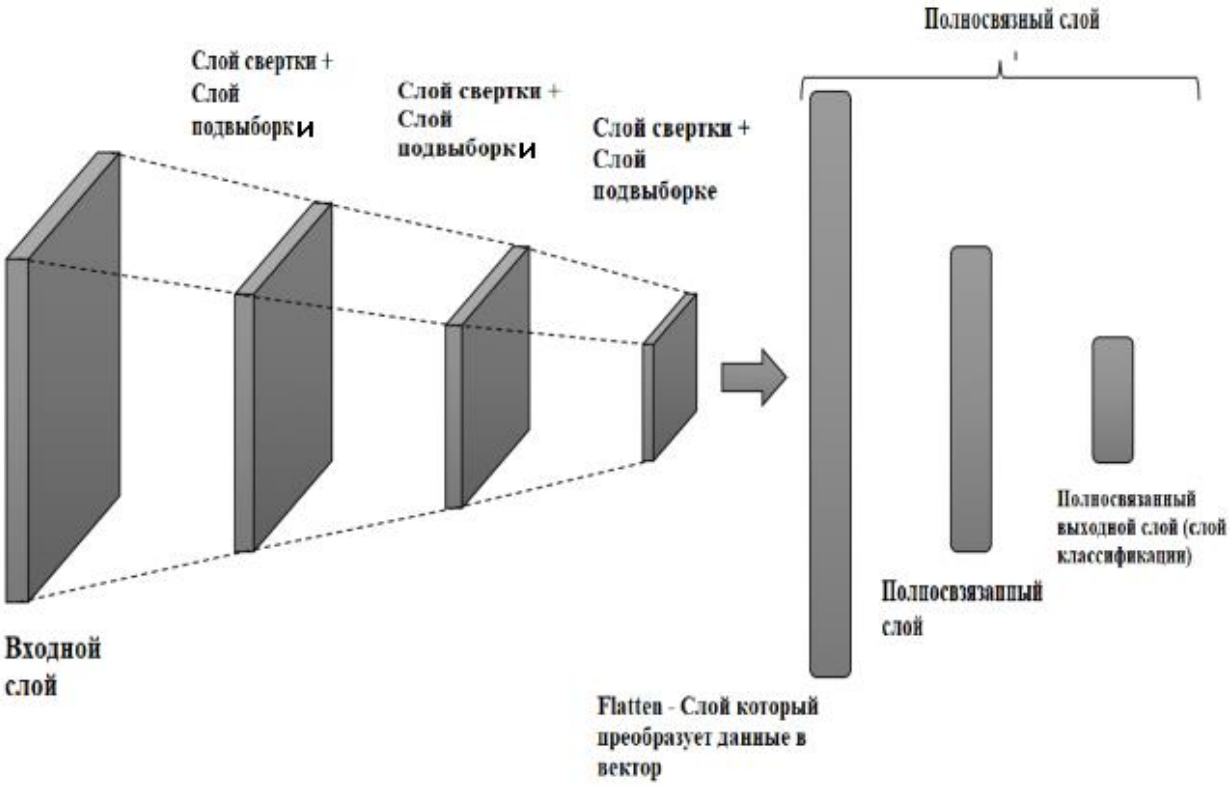

Рисунок 1.5 – Слои сверточных нейронных сетей

Слева изображены слои свертки и подвыборки, они чередуются друг за другом. Потом идет слой преобразования карт признаков полученных из сверточных слоев в вектор, а потом следует слой полносвязного слоя нейронной сети, последний слой размер которого равен количеству классифицируемых объектов дает на выход сигнал активации классов. Самая большая активация класса, которая выдает нейронная сеть, предсказывает нахождение данного объекта в входных данных, входными данными является изображение.

Также карт признаков на слое свертки может быть несколько, и они являются независимыми между собой, то есть имеют своё собственное ядро свертки, ниже будет представлен пример с «независимыми картами свертки».

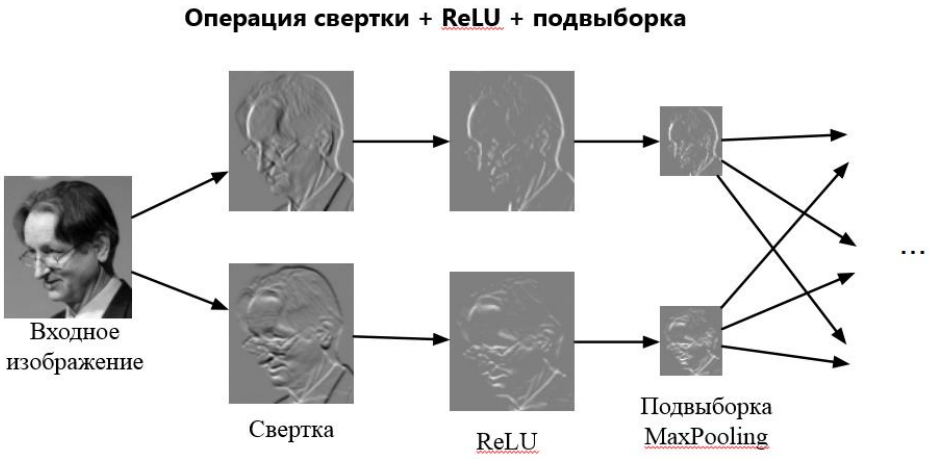

Рисунок 1.6 - Визуализация свертки и подвыборки

На рисунке 1.6 продемонстрирована визуализация свертки и подвыборки, также продемонстрирован результат работы 2 карт признаков, как независимых, то есть имеющих разные ядра свертки, это помогает на каждой карте выделять свои признаки для каждого типа объекта и его положение в пространстве. На рисунке 1.6 также показана функция активации ReLU, которая применяется и в персептроне, описание функций активации, сравнение с другими и визуализация данной функции будет ниже.

Слой под выборки. Слои свертки, которые выдают карты признаков, должны выдавать абстрактные данное об объектах, это позволяет обучать нейронную сеть не на нахождение именно определенного объекта, который был дан нейронной сети на этапе обучения, а поиск абстрактных признаков класса. При этом подход с использованием слоя подвыборки позволяет после операции свертки удалять данные, которые присуще именно данному объекту и ориентироваться на признаки общие классу. Сам слой подвыборке представляет собой проход квадратичной матрицы размеров в 2 ячейки, и после прохода выбирается те данные, что заданы функцией операции под выборки. Выделяют 2 популярные функции под выборки:

- Под выборка поиском среднего значения;

- Под выборка поиском максимального значения.

Под выборка на основе среднего значения представлена в виде формулы ниже.

$$F = mean(X_{i,j} * k_{i,j})$$- функция 1.2 поиска среднего

Под выборка на основе среднего значения представлена в виде формулы ниже.

$$F = max(X_{i,j} * k_{i,j})$$- функция 1.3 поиска максимального

Как писалась выше, квадратичная матрица имеет размер равным 2 ячейкам, что позволяет иметь размер в 2 раза. Опираясь на статью Мин Лина и соавторов «Сеть в сети», была выбрана функция поиска максимального значения. Эта функция позволяет находить более общие признаки, не учитывая и откидывая ненужную информацию в данном слое. Визуальная демонстрация работы функции поиска максимального значения в слое подвыборке продемонстрирована на рисунке 1.7.

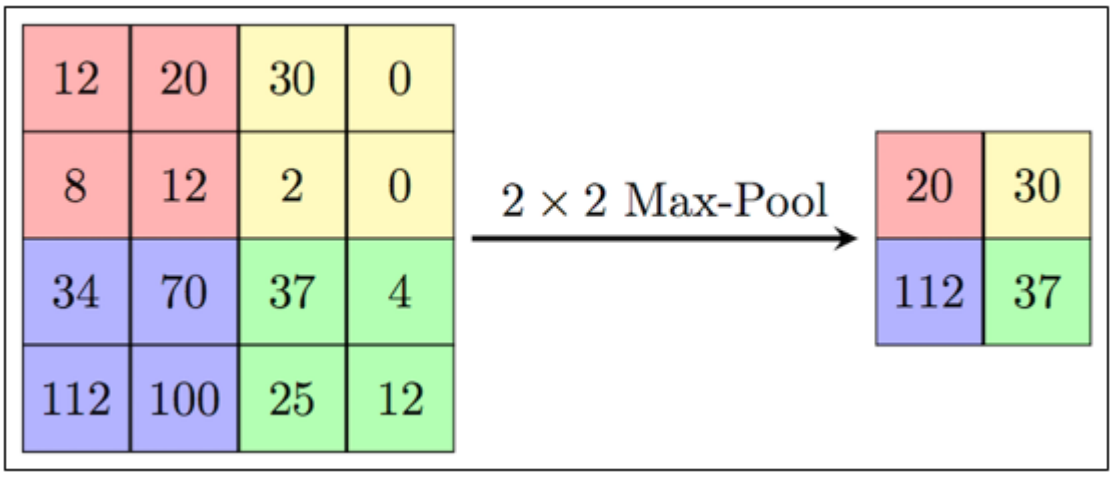

Рисунок 1.7 - Визуальное представление функции поиска максимума слоя подвыборке

На рисунке продемонстрирована квадратичная матрица размером 4 ячейки и матрица слоя подвыборке размеров в 2 ячейке. Видно что матрица выделяет участок слева вверху и ищет максимальное значение, которое равно 20, потом матрица сдвигается размером матрицы на слое подвыборке и ищет

вычисляет следующие значения. В итоге входная матрица уменьшилась наполовину и стала равна размеров в 2 ячейки.

Еще главным шагом при создание архитектуры нейронной сети является выбор функции активации нейрона. Функции активации сверточной нейронной сети и полносвязной нейронной сети является одинаковыми и продемонстрированы ниже:

- Функция активации в виде ступеньки;

- ReLU — Линейный выпрямитель;

- Softmax – Обобщение логической функции.

Первые нейронные сети использовали функцию активации в виде ступеньки, ниже представлена формула этой функции.

$$f(x) = \begin{cases} x < 0 = 0 \\ x \geq 0 = 1 \end{cases}$$ - 1.4 функция активации в виде ступеньки.

Визуализация этой функции представлена на рисунке 1.8.

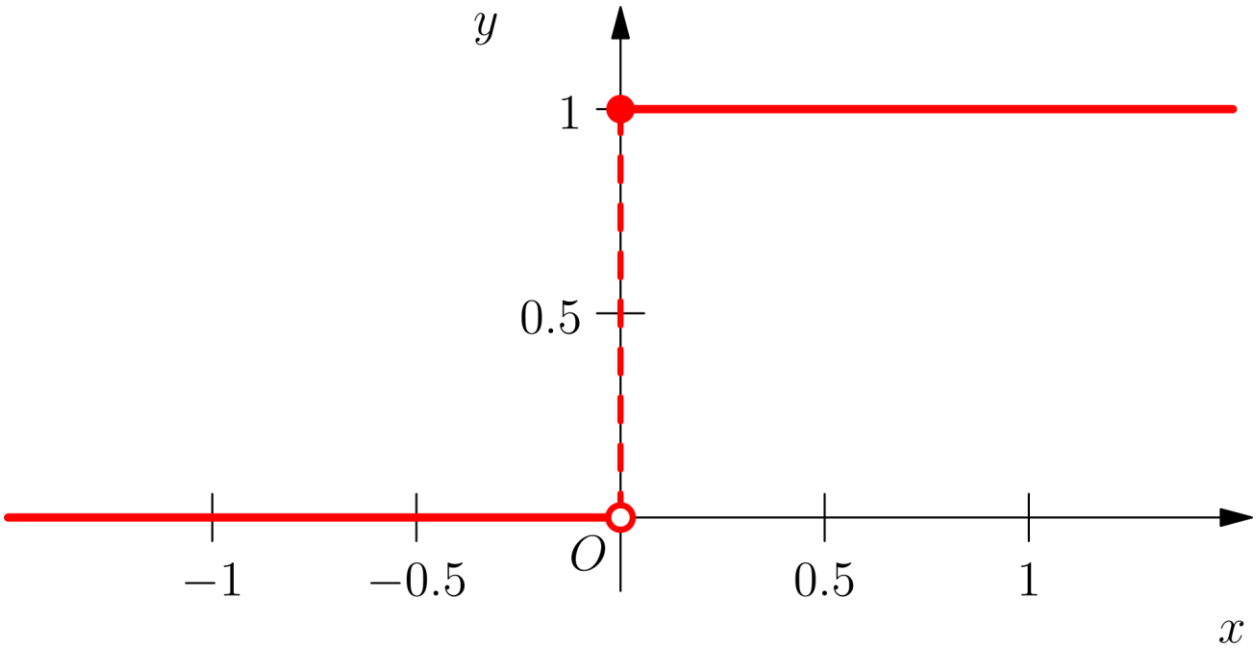

рисунок 1.8 - Функция активации в виде ступеньки

Эта функция, используемая в глубоком обучении может давать предсказания в виде «да» или «нет», и не может выдавать ответы посередине. По причине этим недостаткам её заменили на функцию ReLU, которая может выдвать значения от 0 до 1, то есть «может быть», «может нет» и т. д. Сама функция расшифровывается как Rectified linear unit — Линейный выпрямитель,

$$f(x) = \begin{Bmatrix} x < 0 = 0 \\ x \geq 0 = 1 \end{Bmatrix}$$ - функция 1.5 активации ReLU.

Такая функция дает более точные результаты, по сравнению с функцией ступенчатой. Функция Softmax – используется для классификации нескольких классов объектов, является обобщением логистической функции для многомерного случая. Сама логистическая функция является частным случаем при бинарной классификации, то есть определение между двух классов.

Были рассмотрены нейроны персептрона, полно связной нейронной сети, рассмотрены нейроны сверточной нейронной сети и рассказаны их принципы и разница между ними. Были рассмотрены принципы моделирования «сеть в сети» на основе слоев: сверточный слой, под выборки и полно связанные слои, объяснены их принципы работы. Также были представлены функции активации.

Главным применением сверточнных нейронных сетей является машинное зрение. Машинное зрение — это научное направление в области искусственного интеллекта, в частности робототехники, и связанные с ним технологии получения изображений объектов реального мира, их обработки и использования, полученных данных для решения разного рода прикладных задач без участия (полного или частичного) человека. Главной проблемой в машинном зрении представляет то, как вычислительные устройства воспринимают данные. Если человек видя картинку 1.9 сможет классифицировать объекты которые на ней находятся, то вычислительное

устройство не может этого сделать, т.к. не воспринимает объект как целое а рассматривает его как совокупность пикселей с характеристиками цвета/насыщенностью в зависимости от формата изображения.

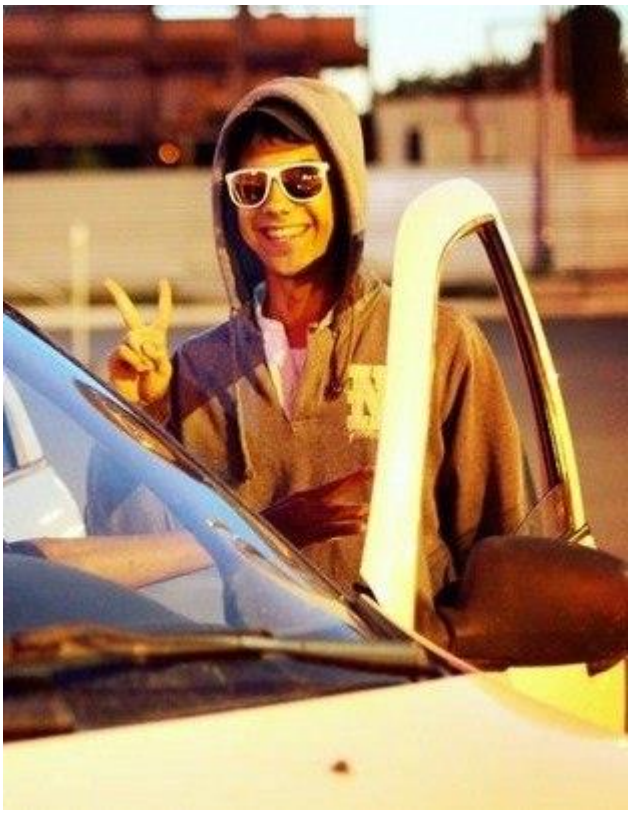

рисунок 1.9 – Пример изображения реального мира

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

рисунок 1.10 – Псевдо пример изображения в компьютерном устройстве

При этом любое изображение в компьютерном устройстве представляет из себя набор пикселей, если речь о черно белом изображении, которые могут принимать значения от 0 до 255, и в зависимости от значения увеличить интенсивность черного цвета, или белого. Но если речь о цветных изображениях, то каждый пиксель представлять из себя либо вектор, представляющий из себя набор цветовых фильтров, популярный из которых является: зеленый, красный и синий. Отсюда видна, что проблема анализа изображения на основе компьютерного представления сложна для человека, и почти не выполнима для компьютера, без сложных алгоритмических просчетов или других математических моделей. Одни из первых способ анализа и обнаружения объектов было удаление максимального шума с изображения, и выделение области, линий объекта и просчета положений этих линий, по ним можно было предположить нахождение объекта, при этом на реализацию поиска одного объекта могли уходить десятилетия и разные ограничения, накладываемые на объект, такие как постоянное местоположение его на изображение.

Проблема машинного зрения сильно актуальна в задачах где необходима автоматизация процесса, исключающего полное или частичное участие человека. Также надо учитывать, что локализация объекта довольно трудоёмкий процесс для разработки, особенно при условии неоднородных объектов.

«Сверточнные нейронные сети» были реализованы, как алгоритм, который решает задачу классификации изображений. Локализация объектов или обнаружение это обощения задача классификации. Так как надо решить сразу две проблемы, это локализация и классификация конкретного объекта на изображении, состоящем из нескольких объектов. Локализация является наиболее сложным алгоритмом компьютерного зрения в виду огромного количества практических случаев. Такаю задачу можно решить несколькими способами, например, использовать метод сквозного окна, требуется разбить

изображение на несколько под изображениями по очереди провести в нейронную сеть, при этом сама нейронная сеть не обязательно должна быть сверточнной нейронной сетью.

Но в данном случае предсказания получатся очень низкого качества, это будет вызвана тем, что на каких изображениях нейронная сеть сможет локализировать объект, а где нет, или сделать совершено ошибочный результат, так же нагрузка вырастет, в зависимости от того, на сколько будет разбито изображение, так как каждое под изображение будет проходить полный цикл в нейронной сети. Разбив изображение на 9 частей, мы вынуждены будем сделать 9 циклов запуска нейронной сети. По этим причинам в современных задачах используют различные модификации сверточнных нейронных сетей. Одним из первый исследователей этой области, Ян Лекун, представил рабочую нейронную сеть, которая решает задачу классификации объектов на изображении.

1.2 Проблемы сверточнных нейронных сетей в задаче локализации объектов

Для определения проблем локализации объектов с использованием алгоритмов глубокого обучения сперва дадим определение Классификации, как главную проблему, которую решают нейронные сети.

Задача классификации - задача, в которой существует множество объектов, которые размечены на некоторое определенное количество классов. Множество объектов является конечным, у которых известно, относиться к какому они классу. Это множество называется «выборкой». Входящие объекты на обучению нейронную сеть не известны. Такую задачу легко решают с применением нейронных сетей. Так как на вход можно подавать данные объекта, например: изображение, текстовый документ и т.д. а на выходе ожидается вектор длиной количеству классов, где каждое выходное значение является вероятностью этого класса в поступающих

данных. Наиболее высокая вероятность является предположительным классом в данных.

Задача локализации – задача нахождения объектов на входных данных, локализация местоположения на данных и возможность классификации. Данная задача не является классической для нейронных сетей и требует построения специфичной архитектуры нейронной сети или использование дополнительных математических алгоритмов. Классическим способом в машинном обучении является разбиение данных на подмножества участков и поочередное прохождение по ним алгоритмом задачи классификации.

Ранее поиск объектов на изображении преимущественно выполнялся при помощи каскадного классификаторов с признаками Хаара или каскадных классификаторов на локальных бинарных шаблонах. Однако с развитием свёрточных нейронных сетей и их успешным применением в области компьютерного зрения каскадные классификаторы постепенно отходят на второй план.

Использование «сверточной сети» обеспечивает малую чувствительность к результатам объектов имеющий искажение на изображении входных данных.

Сформулируем гипотезу исследования.

Гипотезой исследования является предположение, что использование сверточных нейронных сетей может позволить получить точность локализации объектов более 75%, позволит использовать различные датасеты для локальных задач поиска объектов, использование полученных данных может пригодится для практического решения различных проблем машинного зрения.

Таким образом актуальной можно признать цель данного исследования.

Цель исследования – проектирование и разработка программной реализации сверточнных нейронных сетей для задачи локализации объектов

на изображении и повышение скорости работы математической модели за счет использования графических процессоров.

Ниже представлены задачи для решения поставленной цели:

1. Анализ вопроса по теме научного иследования

2. Проектирование математической модели сверточнной нейронной сети и её реализации в виде программного обеспечения для решения проблемы локализации объектов на изображении.

3. Проектирование и реализация программы математической модели для проверки эффективного «сверточной сети».

4. Тестирование математической модели на практике. Получение результатов математических экспериментов и программы.

2 Разработка модели сверточнной нейронной сети для решения задачи классификации объектов на изображении

2.1 Формальное описание задачи локализации объектов на изображении

Задача локализации объектов на изображении напоминает задачу кластеризации объектов, и это не с проста, в обоих случаях надо определить, что за объект с большей вероятностью находиться на изображении. Но в случае с задачей кластеризации, алгоритм довольно прост. Пусть X – это информация подающая на вход нейронной сети, Y – информация полученная в результате работы нейронной сети. В итоге получается уравнение следующего вида:

$$Y = CNN(X) \tag{2.1}$$

X является массивом данных, в случае изображения равен двумерному массиву (ширина, высота). Т.е. представляется в виде набора пикселей. Это проблема того, как хранятся изображения в памяти компьютера и определяющая главную проблему компьютерных наук, «Машинное зрение». При этом сверточнная нейронная сеть должна вернуть предсказания по каждому классу на которых она обучалась.

Например, если нейронная сеть обучалась на английском алфавите, и передачи ей на вход буквы «А», сверточнная нейронная сеть должна вернуть вероятность классов в виде вектора, где, вероятность класса А должна быть выше, чем у остальных классов.

Алгоритм кластеризации сверточнной нейронной сети — это функция, которая получает на вход информацию, а на выходе должна предоставить вектор, на наборе данных классов, на которых она обучалась.

Кластеризация (является обучение без учителя) отличается от классификации (является обучения с учителем) тем, что есть количество классов, но данные при обучении не были разбиты по этим классам, поэтому сверточнная нейронная сеть, должна самостоятельно пытаться найти признаки и пробовать соотносить с заранее известным числом классов. Для

задач локализации и кластеризации при работе с изображениями такой подход категорично не подходит, в этом случае нейронная сеть не сможет научиться находить признаки и кластаризировать их по классам. Единственный вариант использования обучения без учителя, это использовать классов не более 2-х, а сами данные на изображение должны быть максимально отличные друг от друга, например, класс собака и лодка.

Если говорит про задачу локализации, тут все гораздо сложнее, мы заранее не знаем присутствует объект на изображении, а так же не знаем где он находиться, для решения такой задачи, обычно используется следующий подход в большинство современных семейств нейронных сетей, конечный выход нейронной сети разбивается на матрицу, обычно квадратичную, где каждый элемент матрицы представлен следующими параметрами, начало координат объекта, конец координат объекта, вероятность нахождения объекта и вероятность каждого класса. Элемент матрицы в итоге выглядит следующим образом:

$$x1, y1, x2, y2, P(\text{объекта}), class1 \dots classN$$

Где первые значения показывают координаты объекта, на изображении в рамках выделенной области

2.2 Моделирование нейронной сети для классификации изображений

При проектировании математической модели «сверточнной сети» нужно поставить вопрос задачи, условия представлены ниже:

- Поставить задачу которая должна решать нейронная сеть, это может быть классификации, как самая популярная задач, прогнозирование, или модификация;

- анализировать и определить ограничения решаемой задачи «сверхточной сети», такие как, скорость работы сети и её точность в прогнозировании задачи

- определить входные данные, определить тип входных данных и параметров объекта

- определить количество классов, которые являются выходными данными

По теме работы локализации объектов на изображении будем рассматривать задачи только с изображениями. Изображение или данные которые поступают на «сверточнную сеть» состоят из 3 цветов: красный, зеленый, синий. Для того чтоб нейронная сеть смогла с ними работать, надо на входном слое разбить изображение на 3 карты признака- матрицы, каждой карте признака будет соответствовать свой канал цвета, при этом они являются аналогичными входному изображению.

Также после разбивания на каналы цветов, нужно преобразовать значения применив функцию нормализацию, это функция, которая принимает максимальное значение и минимальное, а на выходе нормализует значения в диапазоне от 0 до 1:

$f(x, min, max) = \frac{p-min}{max-min}$ - функция нормализации

В этой формуле f – функция нормализации, x – значение конкретного цвета пикселя от 0 до 255, min – минимальное значение пикселя 0, max – максимальное значение пикселя 255.

$(w, h) = (mW - kW + 1, mH - kH + 1)$ - формула вычисляющая размер выходной карты

Где (w,h) — вычисляемый размер сверточной карты формата ширина и высота, mW – Ширина входной карты, mH – Высота входной карты, kW – ширина сверточного ядра, kH – высота сверточного ядра.

Значения ядра свертки при начале обучения задаются случайным образом числами из диапазона от -0,5 до 0,5. На рисунке 2.8 изображена работа сверточного ядра размером квадратичной матрицы в 3 ячейки, входная карта признаков и выходная карта признаков с размером вычитываемой по формуле выходной карты признаков.

Формула для вычисления квадратичной матрицы выходной карты сверточного слоя приставлена ниже:

$Y = \sum_{w=3,h=3}^{mW,mH} \prod_{k=1,l=1}^{kW,kH} x\,[m-k, n-l] * g[k,l]$ - функция вычисления выходной карты признаков в виде матрицы

Где Y – выходная карта признаков в виде матрицы, x – входная карта признаков в виде матрицы, g – ядро свертки.

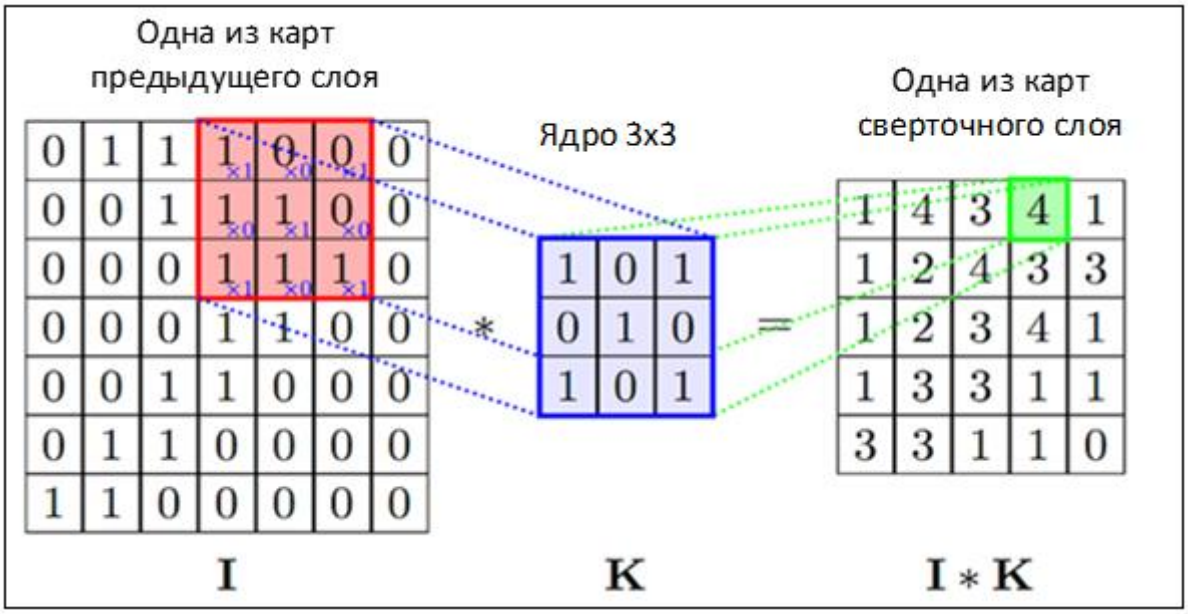

Рисунок 2.8 — операция свертки и получение выходной карты признаков

«Подвыборочный слой». Слой, который помогает сети не переобучение сверточных нейронных моделей является их главным недостатком, так как карты признаков получение на сверточном слое могут обучится так, что активируются только для заданного изображения при обучающей выборке, сам эффект переобучения появляется когда при обучение нейронная сеть показывала хорошие данные, а при проверки на тестовых данных, которая нейронная сеть не видела, показывает значительно худщие результаты, например 20%. Борьба с переобучением планируется следующими методами: использование валидационных данных, метода

прореживания, изменение гиперпараметров, а также использования подвыборочных слоев.

Выбор количества подвыборочных слоев равен количеству сверточных слоев, по принципу чередования, с исключением последнего слоя.

Выходной слой. Количество нейронов соответствует количеству распознаваемых классов, то есть 70 – это значение было задано в начальных данных моделирования нейронной сети. Функция активации этого слоя SoftMax, при анализе литературы эту функцию активации рекомендуется использовать при выходном слое, где задача классификации классов больше 2.

Описав вкратце критерии, была смоделирована следующая сверточная нейронная сеть содержащие 6 слоев сверточной модели, 5 слоев слоя подвыборки, 1 слой персептрона.

В данном разделе была смоделирована сверточная нейронная сеть вида «сеть в сети», это применения как сверточных слоев, так и преобразование сети в векторное пространство и применения слоев полносвязной нейронной сети. Был рассмотрен входной слой который принимает цветное изображение и разбивает его на 3 карты признаков соответствующие каждому каналу цвета: красный, зеленый, синий. Также применена нормализация к значением пикселей к этим картам. Смоделированы и рассмотрены слои свертки, они принимают на вход карту признаков, ядро сверточной сети постепенно проходит по всей входной карте признаков и на выходе получается выходная карта признаков. После применяется функция активации ReLU, которая относит найденные абстрактные признаки к конкретным классам. Также смоделированы слои подвыборке на основе поиска максимального значения, для того чтоб выделять только важные абстрактные признаки для передачи в более глубокие слои нейронной сети.

2.3 Проблема переобучения и методики их решения

Переобучение - это явление, когда обучаемая модель хорошо распознает примеры из обучающего множества, но при этом не распознает или плохо распознает любые другие примеры, не участвовавшие в процессе обучения (т.е. предъявляемые ей в процессе практического использования).

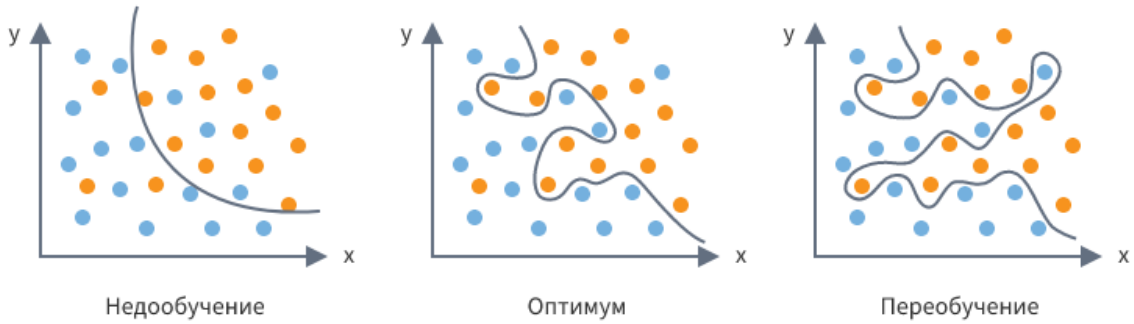

Рисунок 2.10 – пример переобучения на примере кластеризации

На рисунке 2.10 показан пример переобучения алгоритма на обучающих данных при задачи кластеризации. Переобучение — это результат чрезмерной подгонки параметров модели к зависимостям, содержащимся в обучающем множестве. Если происходит переобучение, то модель не приобретает способности к обобщению — возможности распространять обнаруженные на обучающем множестве зависимости и закономерности на новые данные. Такая модель на практике оказывается бесполезной, даже если её ошибка обучения мала.

Проблема переобучения в той или иной степени характерна для всех видов обучаемых моделей, но особенно остро она стоит для нейронных сетей. В процессе обучения производится подгонка весов нейронной сети таким образом, что сеть преобразовывала входы к желаемым выходам в соответствии с зависимостями, обнаруженными в обучающих данных.

Однако, если сеть обучать слишком долго или с использованием слишком большого числа параметров (весов), то произойдёт переобучение. Это связано с тем, что с определённого момента сеть начинает "подстраиваться" не под общие зависимости в данных, а под особенности отдельных примеров, которые могут содержать аномальные значения, ошибки и т.д.

Как следствие, сеть начнёт проверять новые, предъявляемые ей наблюдения, не на соответствие зависимости, а на соответствие отдельным примерам из обучающего множества. В итоге модель сможет распознать новое наблюдение только в том случае, если оно совпадёт с одним из обучающих примеров. Стоит отметить, как описывалось в 1 главе сеть перцептрона горазда более подвержена переобучению, чем сверточная нейронная сеть. Причина: Нейрон и его вес сети перцептрон при работе с изображениями привязывается к пикселю, и в процессе обучения запоминает его интенсивность, полностью игнорируя соседних нейронов, даже использую методологию глубокого обучения, скрытых слоев, остается факт, что нейрнон обрабатывает свой пиксель, и при довольно сильном искажении первоначального изображения, можно наблюдать неправильные предсказания на новых данных, которые отсутствовали при обучении. Нейрон сверточнной нейронной сеть также привязывается к пиксели, но захватывает область в зависимости от размера карты сверточнного слоя. Это помогает анализировать не пиксель на изображении, а определённую область, что в итоге формируют модель, которая менее подвластная проблеме переобучения, но это не значит, что проблема решается.

Для того, чтобы избежать переобучения, можно использовать следующие несложные правила.

1. Применение тестового множества. Тестовое множество формируется из обучающего набора данных случайным образом и его примеры подаются на вход сети между обучающими примерами, но корректировку весов сети на вызывают. Вначале обучения ошибка и на

обучающем, и на тестовом множестве уменьшаются. Но начиная с какого-то ошибка на тестовом множестве начинает расти. Это сигнализирует о начале переобучения и необходимости принудительно остановить процесс обучения.

2. Использование перекрёстной проверки. Все данные, на которых строится модель, разделяются на k блоков равного размера. При этом обучение производится на k-1 блоках, а тестирование - на k-м. Процедура повторяется k раз, при этом для тестирования каждый раз выбирается другой блок. В результате все блоки оказываются используемыми и как обучающие, и как тестирующие.

3. Выбор конфигурации нейронной сети. Нужно выбирать конфигурацию сети (число слоёв и нейронов), чтобы количество параметров модели (весов) была в 2 - 3 раза меньше числа примеров обучающего множества. Если число параметров модели и обучающих примеров окажется соизмеримым, то сеть просто "запомнит" все комбинации вход-выход в примерах обучающего множества, и будет воспроизводить только их, а на новых данных допускать ошибки.

4. Метод dropout. На сегодняшний день является одним из самых эффективных способов борьбы с проблемой переобучения. Имеет два варианта, выключение скрытого слоя в момент обучения сети, рассматриваться не будет, по причине сложной реализации. Схема ниже иллюстрирует данный метод. Работу приема показана на рисунке 2.11.

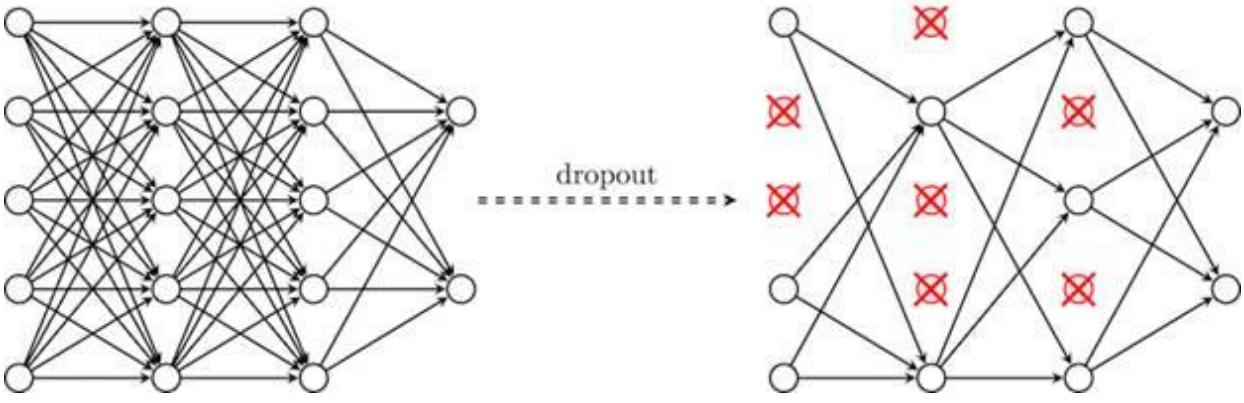

Рисунок 2.11 – пример выполнения dropout для нейронной сети

Также стоит выделить самый эффективный способ борьбы с переобучением, это использование расширение обучающий выборки, если выполняю условие 3, мы не будем формировать постоянно новую выборку, при условии, что мы используем свой собственный датасет, то точность модели нельзя будет поднять, за счет высоко снижения процента переобучения модели сети.

2.4 Выбор модели сверточнной нейронной сети для проведения экспериментов

Сверточнные нейронные сети были реализованы, как алгоритм, который решает задачу классификации изображений. Локализация объектов или обнаружение это обощения задача классификации. Так как надо решить сразу две проблемы, это локализация и классификация конкретного объекта на изображении, состоящем из нескольких объектов. Локализация является наиболее сложным алгоритмом компьютерного зрения в виду огромного количества практических случаев. Такаю задачу можно решить несколькими способами, например, использовать метод сквозного окна, требуется разбить изображение на несколько под изображениями, как на рисунке 2.9, и по очереди провести в нейронную сеть, при этом сама нейронная сеть не обязательно должна быть сверточнной нейронной сетью.

Рисунок 2.9 — Применение способа скользящего окна

Но в данном случае предсказания получатся очень низкого качества, это будет вызвана тем, что на каких изображениях нейронная сеть сможет локализировать объект, а где нет, или сделать совершено ошибочный результат, так же нагрузка вырастет, в зависимости от того, на сколько будет разбито изображение, так как каждое под изображение будет проходить полный цикл в нейронной сети. Разбив изображение на 9 частей, мы вынуждены будем сделать 9 циклов запуска нейронной сети. По этим причинам в современных задачах используют различные модификации сверточнных нейронных сетей. На рисунке 2.10 показаны популярные модификации сверточнных нейронных сетей способных показывать хорошие результаты при выполнении задачи локализации объектов.

Рисунок 2.10 — Виды нейронных сетей для локализации объектов

Результаты были взяты с проекта deepsystem. Для реализации была выбрана сверточнная нейронная сеть модификации YOLO. Данная модель обладает Singleshoot реализацией – однопроходной, это значит, что выходные данные нейронной сети будут сделаны за один проход. Таким свойством обладают нейронные сети SSd и YOLO. Ниже описаны кратно виды сверточнных нейронных сетей:

1. R-CNN - первая модель для решения задачи локализации объектов. Работает как обычный классификатор изображений. На вход сети подаются разные регионы изображения и для них делается предсказания. Очень медленная так как прогоняет одно изображение несколько тысяч раз;

2. Fast R-CNN - Улучшенная и более быстрая версия R-CNN, работает по похожему принципу, но сначала все изображение подается на вход CNN, потом из полученного внутреннего представления генерируются регионы. Но по-прежнему довольно медленная для задач реального времени.

3. Faster R-CNN - Главное отличие от предыдущих в том, что вместо selective search алгоритма для выбора регионов использует нейронную сеть для их «заучивания».

4. SSD. Вариация YOLO, но в качестве сети для извлечения признаков использует VGG16. Относительно быстрая и пригодная для работы в реальном времени.

5. YOLO - Имеет совсем другой принцип работы по сравнению с сетями R-CNN, не использует регионы. Имеет огромное количество вариаций архитектур, некоторые могут выигрывать в скорости у нейронных сетей SSD.

Еще хочется отметить две нейронные сети, которые тоже способны справляться с этой задачей. Они не имеют огромную популярность, как предыдущие:

6. Feature Pyramid Networks (FPN) - Разновидность сети типа Single Shot Detector, из-за особенности извлечения признаков лучше, чем SSD распознает мелкие объекты.

7. RetinaNet - Использует комбинацию FPN+ResNet и благодаря специальной функции ошибки (focal loss) дает более высокую точность (accuracy).

Выше уже были сделаны выводы, почему для реализации была выбрана сверточнная нейронная сеть YOLO. Поговорим про неё более подробно. You Only Look Once[11] — это очень популярная на текущий момент архитектура CNN, которая используется для распознавания множественных объектов на изображении. Главная особенность этой архитектуры по сравнению с другими состоит в том, что большинство систем применяют CNN несколько раз к разным регионам изображения, в YOLO CNN применяется один раз ко всему изображению сразу. Сеть делит изображение на своеобразную сетку и предсказывает bounding boxes и вероятности того, что там есть искомый объект для каждого участка, пример показан на рисунке 2.11.

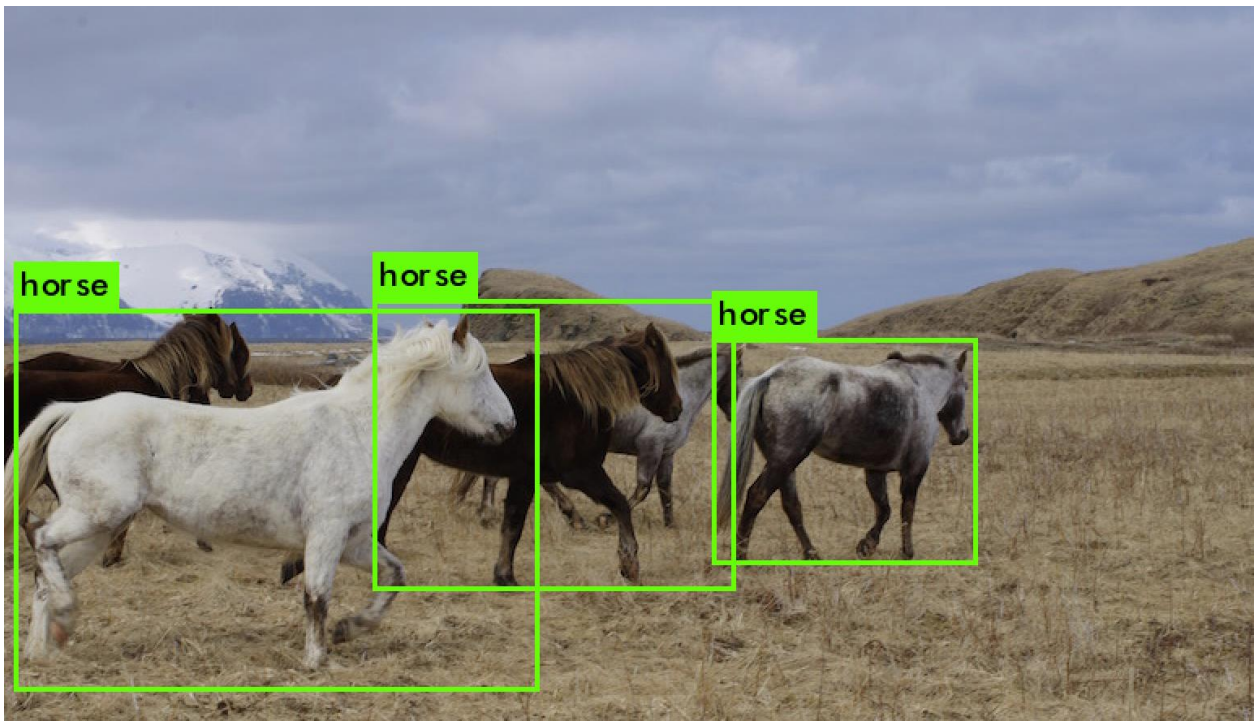

Рисунок 2.11 – Пример предсказаний выдаваемых нейронной сетью

Плюсы сверточнных нейронных моделей YOLO состоит в том, что сеть смотрит на все изображение сразу и учитывает контекст при детектировании и распознавании объекта. Так же YOLO в 1000 раз быстрее чем R-CNN и около 100 раз быстрее чем Fast R-CNN. Для большего ознакомления про скорость сверточнной нейронной сети YOLO можно ознакомиться в статье в YOLOv3: постепенное улучшение[20].

Модель YOLOv3. Это усовершенствованная версия архитектуры YOLO. Она состоит из 106-ти свёрточнных слоев и лучше детектирует небольшие объекты по сравнению с её предшественницей YOLOv2. Основная особенность YOLOv3 состоит в том, что на выходе есть три слоя каждый из которых рассчитан на обнаружения объектов разного размера. На рисунке 2.12 приведено её схематическое устройство.

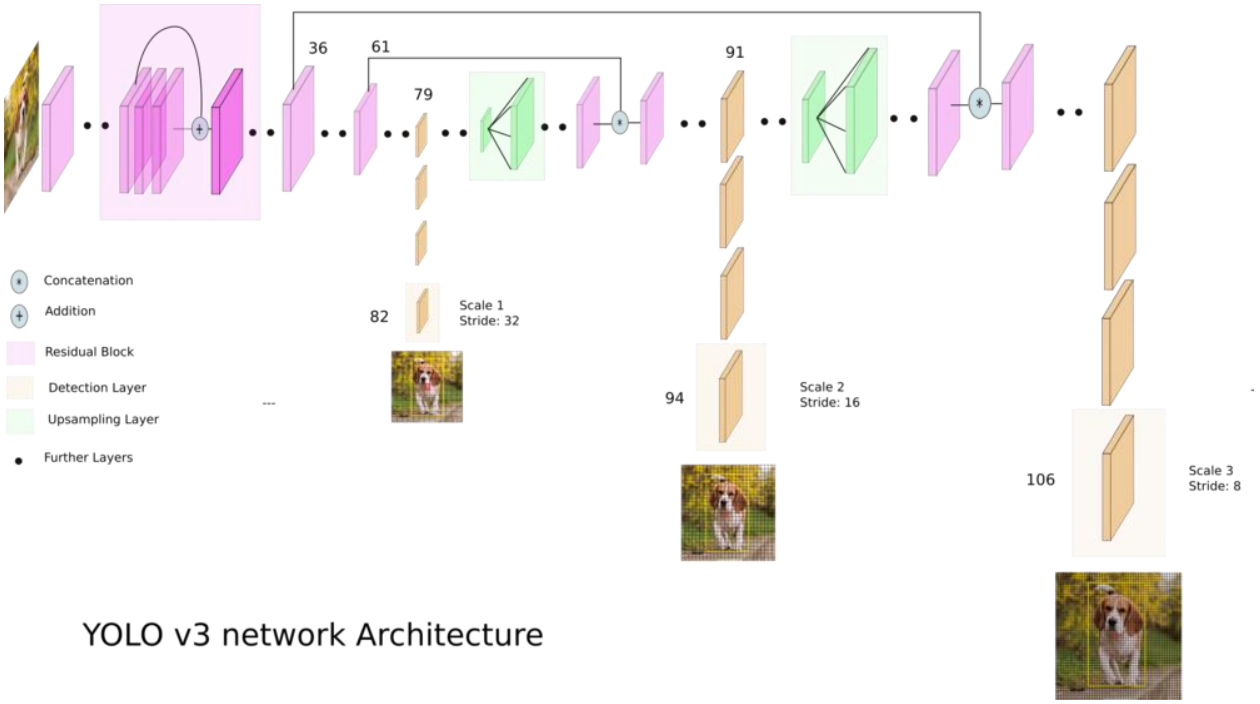

Рисунок 2.12 – Архитектура и принцип сверточнной нейронной сети YOLOv3.

Для научно исследовательской работы данная модель является крайне избыточной и медленной, тем более для задач локализации объектов на изображении в реальном времени, поэтому была выбрана модификация сверточнной нейронной сети YOLOv3-tiny. Это обрезанная версия архитектуры YOLOv3, состоит из меньшего количества слоев (выходных слоя всего 2). Она более хуже предсказывает мелкие объекты и предназначена для небольших датасетов, что является огромным преимуществом, так как датасеты является огромной проблемой в задачи нейронных сетей. Из-за урезанного строения, веса сети занимают небольшой объем памяти, что может сыграть роль на менее производительном железе и она выдает хороший показатель «количество кадров секунду». Именно такие преимущества и были сделаны в выборе модели сверточнной нейронной сети. Стоит отметить что были сделаны следующие модификации, был убран промежуточный слой выборки. Было уменьшено общее количество сверточнных нейронных слоев + слоев подвыборки. Но были добавлены промежуточные сверточнные слои перед слоем под выборки.

2.5 Результаты вычислительных экспериментов с применением сверточнных нейронных сетей

Первой реализацией была сверточнная нейронная сеть, которая пыталась локализировать на изображении фигурки конструктора Lego. Выбор такого датасета был связан с уже готовой реализацией сверточнной нейронной сети для задачи классификации. Пример одной картинки из датасета представлен на рисунке 2.13.

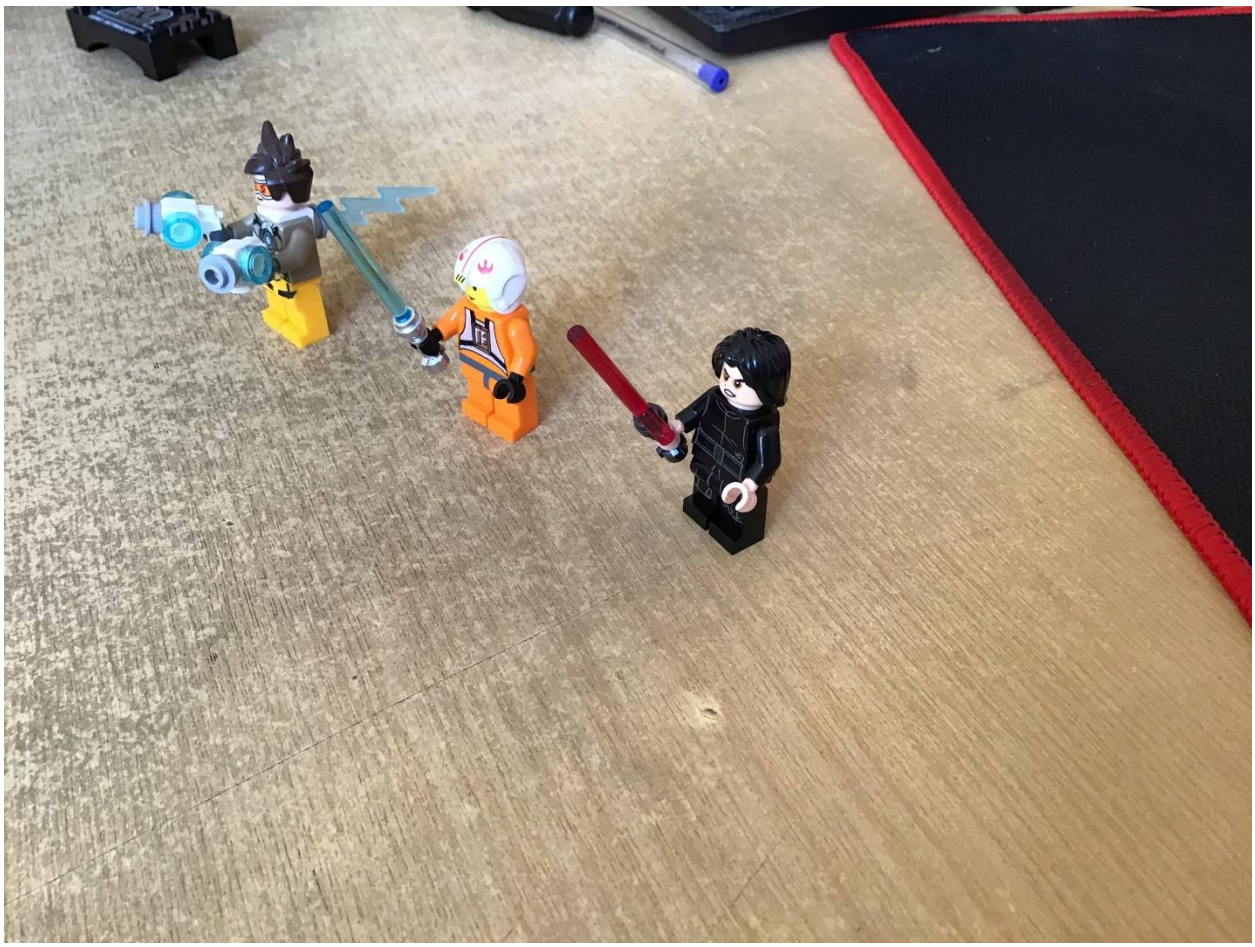

Рисунок 2.13 – пример датасета Lego минифигурки

Для обучения были сгенерированы метки, содержащие следующие атрибуты: метку класса, расположение объекта, расположение файла. Пример содержания файла представлен на рисунке 2.14. Для ручного составления этих меток был использован инструмент Label-Annotation-VOC-Pascal[21]. Он написан на языке Python. Принцип работы заключался в нанесении рамки на изображение. Метки выделенному объекту. Пример работы этой программы показан на рисунке 2.14.

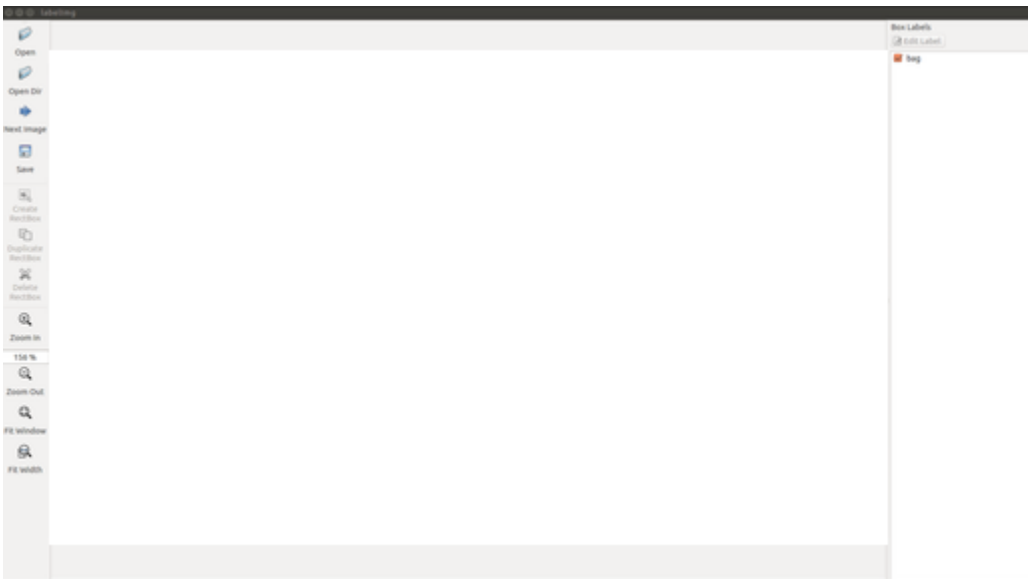

Рисунок 2.14 – Программа Label-Annotation-VOC-Pascal

К сожалению, данный датасет содержал 33 изображения, и после обучения сверточнной нейронной сети, при условии, что веса были инициализированы случайным образом. Самая высокая точность нахождения локализироваемого объекта на изображении не превышала 20%, причина была в недостаточном количестве и качестве данных для обучения. При этом самостоятельное создание датасетов, даже состоящие из малого количества занимает огромное количество времени. Было принято решение использовать открытые источенные датасетов, такие как: Pascal Voc 2007[22], Pascal Voc 2014 и COCO dataset[23]. При анализе этих датасетов был выбран только первый, по причине достаточны количество изображений и меток. COCO был отброшен по причине неправильной организации аннотации к классам.

```
<annotation>
        <folder>jpegmini_optimized</folder>
        <filename>001.jpg</filename>
        <path>C:\Users\Yagorka\Downloads\jpegmini_optimized\001.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>3264</width>
                <height>2448</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>lego_minifigure</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>612</xmin>
                        <ymin>878</ymin>
                        <xmax>1086</xmax>
                        <ymax>1375</ymax>
                </bndbox>
        </object>
        <object>
                <name>lego_minifigure</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>1404</xmin>
                        <ymin>875</ymin>
                        <xmax>1766</xmax>
                        <ymax>1339</ymax>
                </bndbox>
        </object>
        <object>
                <name>lego_minifigure</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>2025</xmin>
                        <ymin>857</ymin>
                        <xmax>2371</xmax>
                        <ymax>1319</ymax>
                </bndbox>
        </object>
</annotation>
```

Рисунок 2.17 – Пример аннотация для файла из датасета Lego

CUDA – «программно-аппаратная архитектура параллельных вычислений, которая позволяет существенно увеличить вычислительную производительность благодаря использованию графических процессоров фирмы Nvidia». На рисунке 2.18 показано как центральный процессоров использует средства графического процессора для обработки информации.

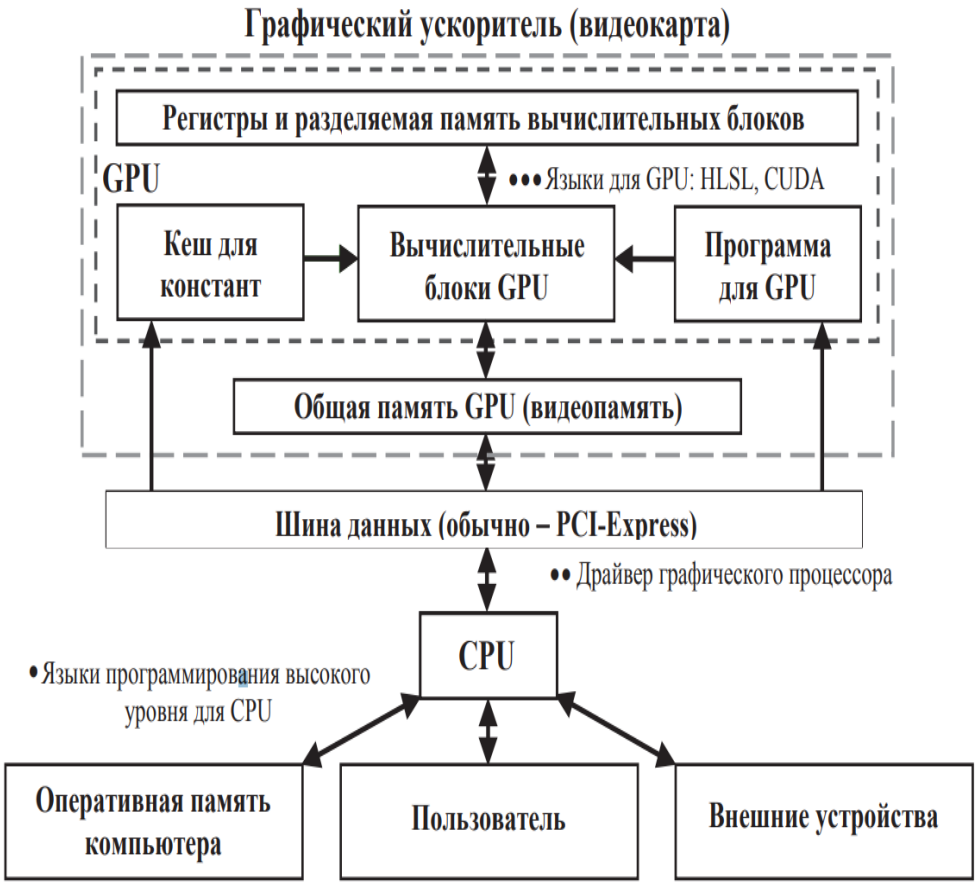

Рисунок 2.18 — Схема взаимодействия CPU и GPU с памятью и между собой

Выбор CUDA обусловлен такими факторами:

- Наличие графического процессора производством Nvidia поколения Pascal;

- Поддержка CUDA на языке Python с использованием сторонних библиотек.

Теоретическое обоснование. Сверточная нейронная сеть на уровне локального представления представляет из себя матрицу, это расположение пикселей. Набор таких слоев равен количеству выходных нейронов слоя, умноженный на уровень слоев нейронной сети. Слои перцентрона (полносвязанного слоя) представляют из себя вектор. Эти данные подвергаются простым математическим действиям. При параллельной обработки матрицы или вектора можно получить выигрышь в скорости

работы нейронной сети N раз, где N – количество потоков в программе. Современные процессоры ограничены до 4-12 потоков, а графические процессоры имеют гораздо больше параллельных потоков обработки информации. Использую параллельные вычисления на графическом процессоре при работе или обучение нейронных сети, теоретически должны получить производительность в несколько раз при тех же вычислениях на процессоре.

Результатом внедрения CUDA и работе нейронной сети на графическом процессоре GP106, Palit Jetstream 1060 6Gb Nvidia представлены ниже с результатами на центральных процессоров i5 и i7 производства Intel.

Результат работы сверточной нейронной сети с использованием процессора i5 поколения Ivy Bridge, 2 ядра, 4 потока – менее 0.5 кадров в секунду,

Результат работы сверточной нейронной сети с использованием процессора i7 поколения coffee lake, 6 ядер, 12 потоков — менее 1 кадра в секунду.

Результат работы сверточной нейронной сети с использованием графического процессора GP106, количество CUDA ядер 1280 — скорость обработки видео потока более 23 кадра в секунду.

Прирост составил более чем 23 раз при обработки видео потока. Результаты показаны на рисунке 2.19.

Библиотека используемая на языке Python, была написано на C языке, и под собой использует язык программирования CUDAC, для которого требуется установка специальных библеотек перечисленных ниже для компиляции и запуска приложений на графическом процессоре:

- cuDNN, специально разработана для обучения глубоких нейронных сетей. Она содержит оптимизированные для GPU реализации сверточных и рекуррентных сетей, различных функций активации. cuDNN позволяет обучать нейронные сети на GPU в несколько раз быстрее, чем просто CUDA;

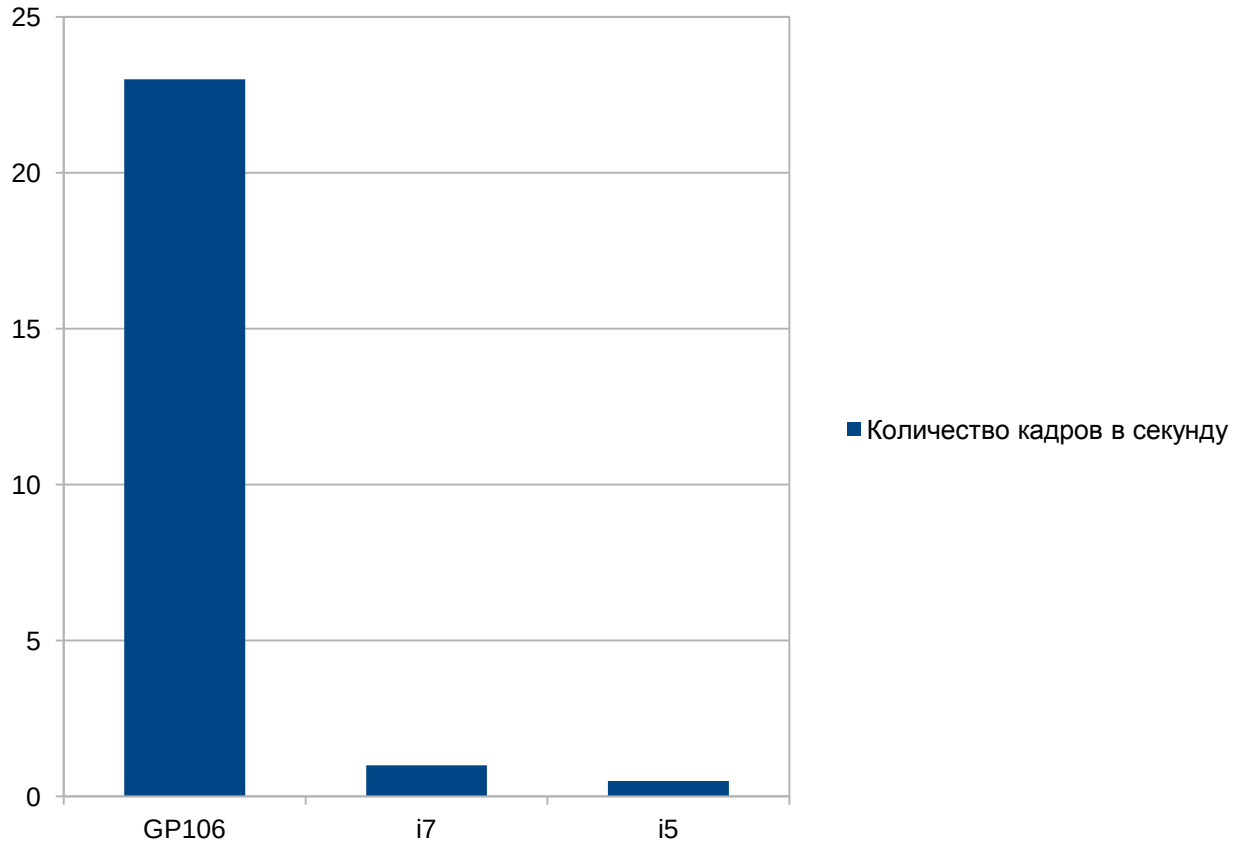

Рисунок 2.19 — Диаграмма скорости обработки видео потока на графическом процессоре и центральном процессоре.

Так как скорость обработки с использованием графического процессора GP106 увеличился в 23 раза по сравнению с центральным процессорам i7, результат переноса обработки алгоритмов нейронной сети признается успешным, хотя теоретически 1280 CUDA ядер должны были быть в 106 раз быстрее, чем 12 потоков центрального процессора. Это было получено из формулы nCUDA / nCPU, где nCUDA количество CUDA ядер графического процессора, nCPU – количество ядер центрального процессора. Возможные потери в производительности по сравнения с теоретическими были вызваны, тем, что использовались библиотеки на языке Python, а также вызваны обменом данными между центральным процессором и графического процессора, так как в теоретических данных не были учтены эти данные.

В результате использования графического процессора были достигнуты показатели в виде увлечения скорости 23 раз, чем на центральном процессоре. Результат исследования применения графического процессора полностью удовлетворяет ожидания.

Также проводились эксперименты с использованием HOG алгоритма, как замена сверточнной нейронной сети. Статистика всех данных представлена на рисунке 2.20.

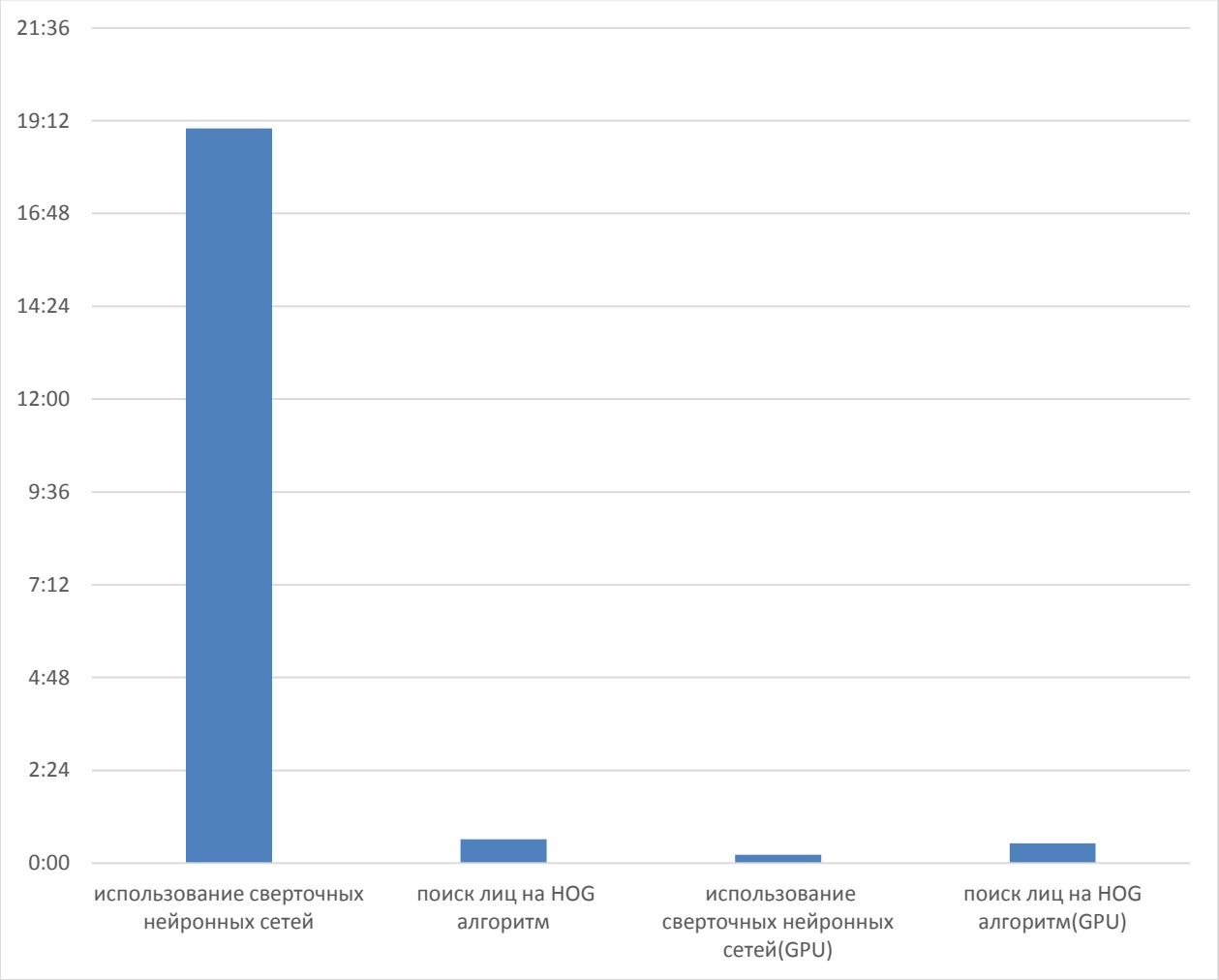

Рисунок 2.20 — Диаграмма скорости обработки видео потока на графическом процессоре и центральном процессоре.

Как видно, сверточнные нейронные сети являются не самым оптимальным алгоритмом машинного обучения и использование HOG алгоритма для поиска объектов гораздо эффективнее справляется с задачей локализации объектов. Но если использовать мощности графического процессора, то ситуация меняется, и сверточнная нейронные сети выигрывают по времени более, чем в два раза, это сказывается использование однородных математических действий, и собственно эффективность их распараллеливания [21]. Ниже представлены данные:

При использование сверточных нейронных сетей время выполнения составило примерно 19 часов. Время начала 12:27:18. Время окончания 07:14:35

Заменяем поиск лиц на HOG алгоритм - время примерно 37 минут. Время начала 11:29:15. Время окончания 12:07:00

При использовании сверточнных нейронных сетей(GPU) время выполнения составило примерно 13 минут. Время начала 20:57:50. Время окончания 21:11:37

Заменяем поиск лиц на HOG алгоритм(GPU) - время примерно 31 минут. Время начала 21:15:11. Время окончания 21:46:24

3 Программная реализация

3.1 Описание разработанного программного обеспечения

В ходе проведение описанных в данной работе исследований было разработано программное обеспечение в виде программы с работой по видеопотоку и локализации объектов через камеру с количеством распознаваемых классов 20 штук. Также на основе ПО было разработано следующие программное обеспечение в виде клиент и сервер, при этом разработка клиента велась в виде java программы, для персональных машин, в виде react для интернета клиентов (браузера) и в виде react native (для мобильных устройств в виде приложения), обладающее следующими функциональными возможностями:

1) Загрузка фотографий для обработки или передача видеопотока для покадровой обработки;

2) Выполнение задачи локализации объектов на изображении или видеопотоке с возможностью графической отображения рамок местоположения объектов и подписи найденного класса;

3) Ведение лога выполнения задачи для дальнейшех обработки и анализа полученных данных;

4) Заранее продуманная архитектура сверточных нейронных сетей для добавления дополнительного функционала в дальнейшее развитие программного продукта;

5) Возможность производить вычисления на графических процессорах(GPU) компании NVIDIA, благодаря поддержки технологии cuda, что позволяет ускорять работы нейронных сетей;

6) Возможность экспортирования логов работы программного обеспечения для дальнейшего анализа в формате файла txt.

3.2 Алгоритм работы с приложением

Запуск программного продукта по обработки видеопотока для задачи локализации объектов приведет к открытию основного окна программы (рисунок 3.1). Данное окно состоит из следующих элементов управления, доступных пользователю:

- Отображения работы сверточнных нейронных сетей для задачи локализации объектов. Помешенные найденных объектов в рамки и подписи класса.

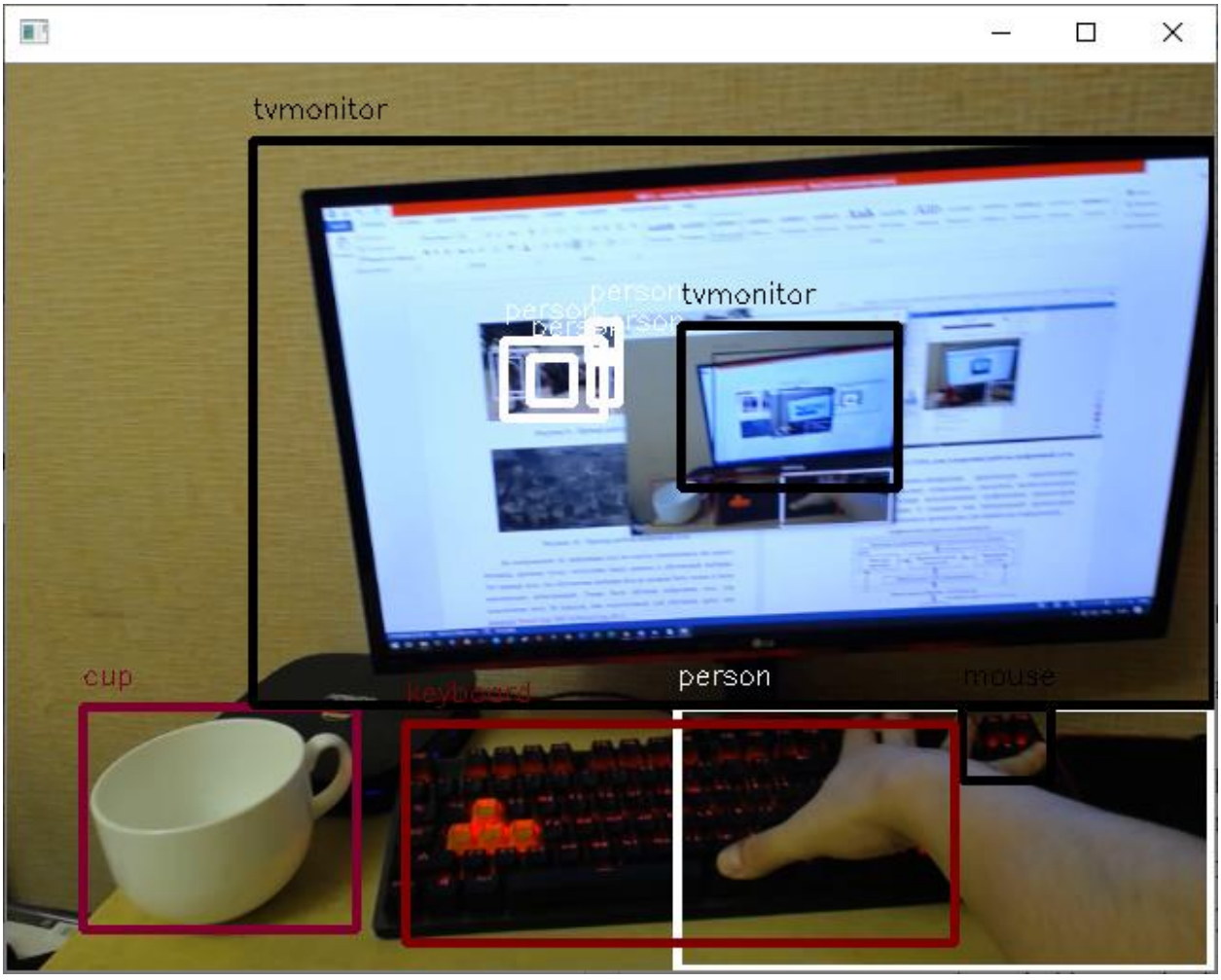

Рисунок 3.1 – Основная форма разработанной программы

Программа работает на обученных нейронных сетей по локализации объектов, количество локализуемых объектов 20 штук, датасет для обучения использовался Pascal VOC2012 и Pascal VOC2014.

Для представленная практического применения сверточной нейронной сети, был реализован проект «Идентификация человека по лицу». Абстрактно проект включает в себя три метода машинного обучения. Алгоритм работы является следующим, исходное изображения поступает на сверточнную нейронную сеть, которая занимается задачей локализации объекта, нужно выделить фронтальное лицо человека указав его местоположение, также решается еще одна задача, если на изображение будет несколько групп лиц, то для каждого найденного будет найдено своё местоположение. Далее полученные под изображения попадают в следующие сверточнную нейронную сеть, обученную находить точки на лице человека, количество искомых точек равняется 64, так как она была обучена на датасете, содержавшем именно такое количество аннотируемых данных. Для увеличения точности, можно использовать другой датасет, содержащий еще более количество аннотируемых точек на лице. Потом полученный вектор из этой сети размерности 128 элементов (64 точки умноженные на 2, x и y), поступают на алгоритм Эвклидова расстояния, при этом выходное значения сравнивается с заданным числом сравнения. В моем проекте это число 0,6. Если выходное число является на промежутке от 0 до 0,6 включительно, то алгоритм делает предположение, что люди равны, если больше, то это разные люди. Для примера работы данного алгоритмы, была написана программа на языке Python, которая берет видеоизображение, так же получает папку, содержащею людей, которые предположительно есть в данном видеоизображении. На рисунке 12 показаны актеры фильма «волк с уолл стрит». Данные фотографии были взяты с сайта кинопоиск, также эти фотографии не учувствовали в обучающей выборке. Результаты работы алгоритма представлены на рисунках 3.3-3.5.

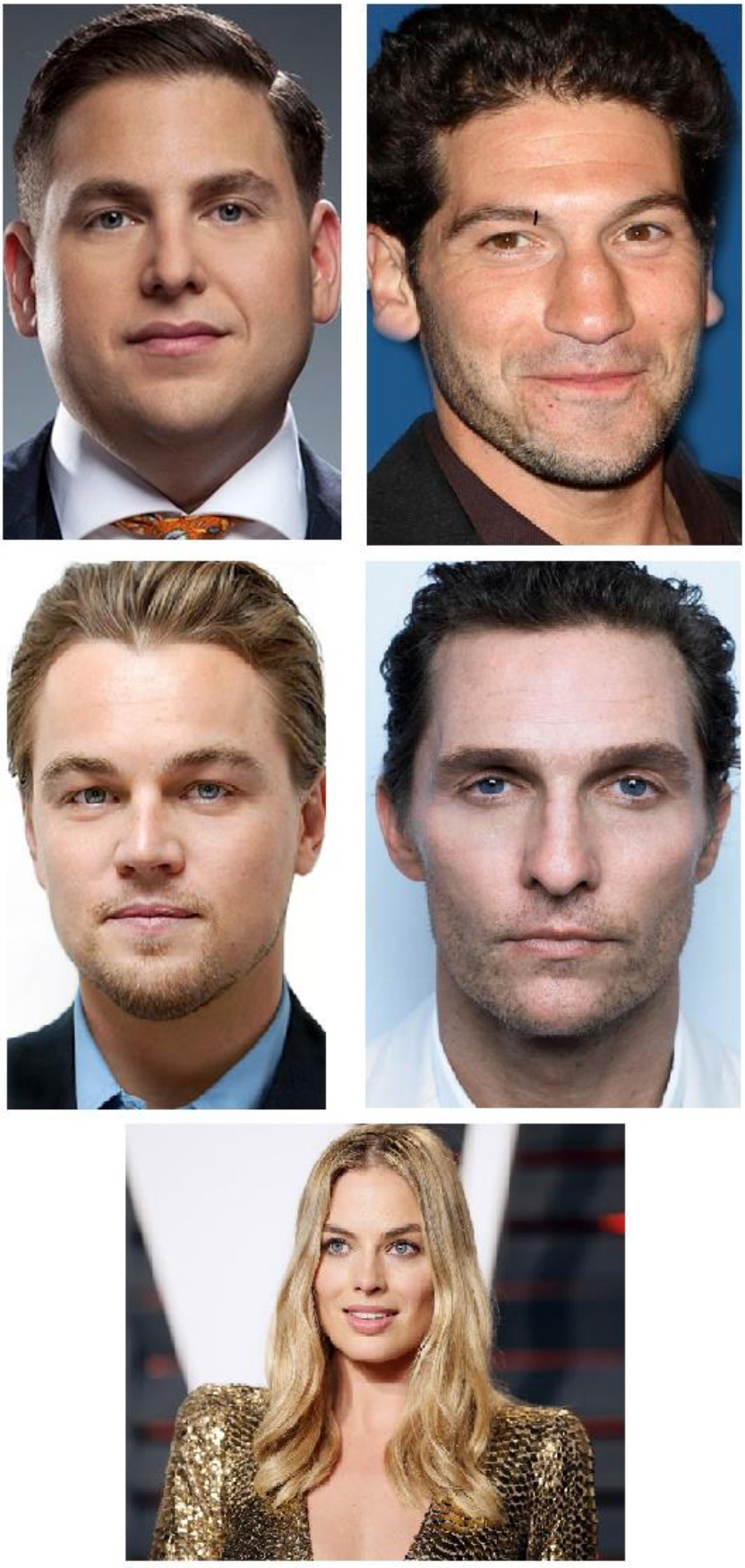

Рисунок 3.2 – актеры фильма «волк с уолл стрит»

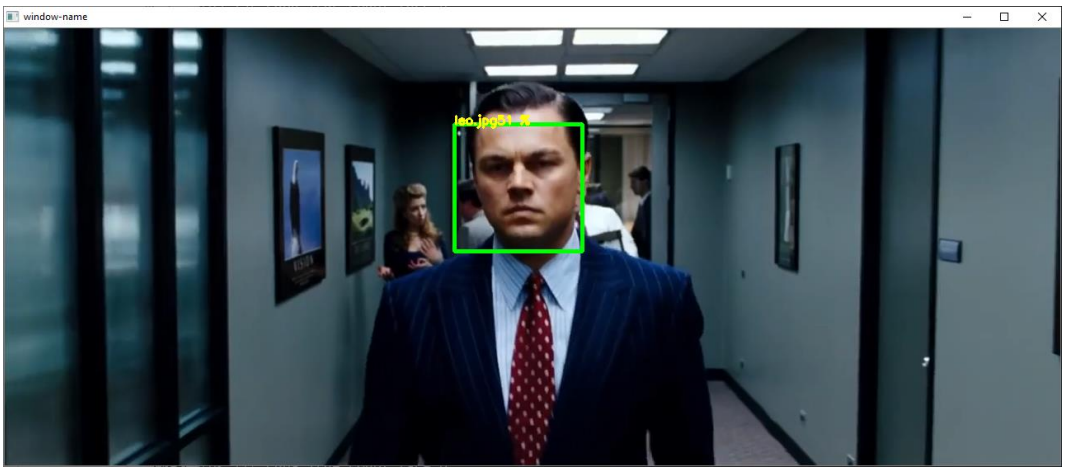

Рисунок 3.3 – пример работы алгоритма

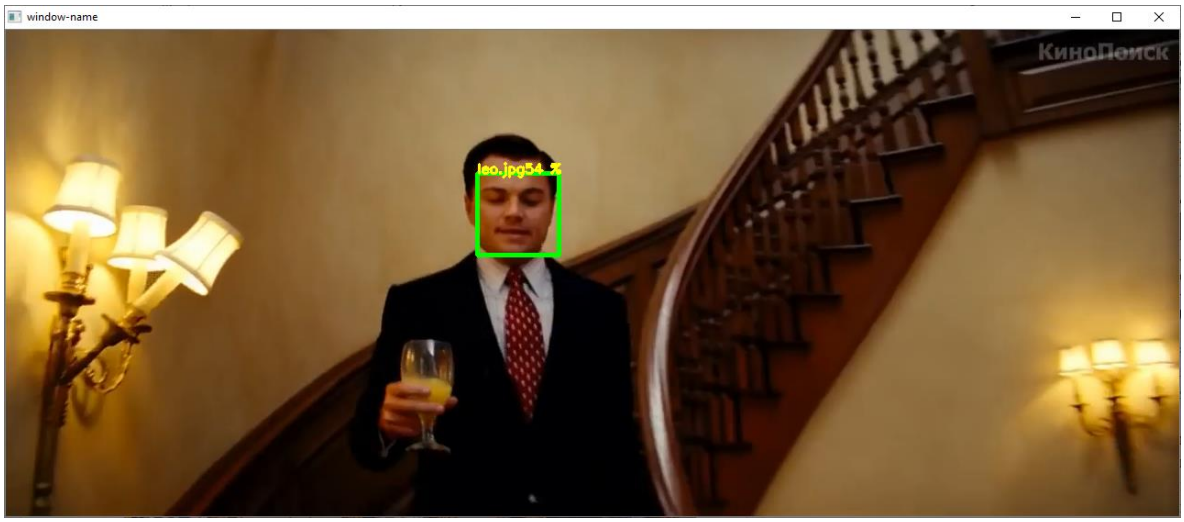

Рисунок 3.4 – пример работы алгоритма

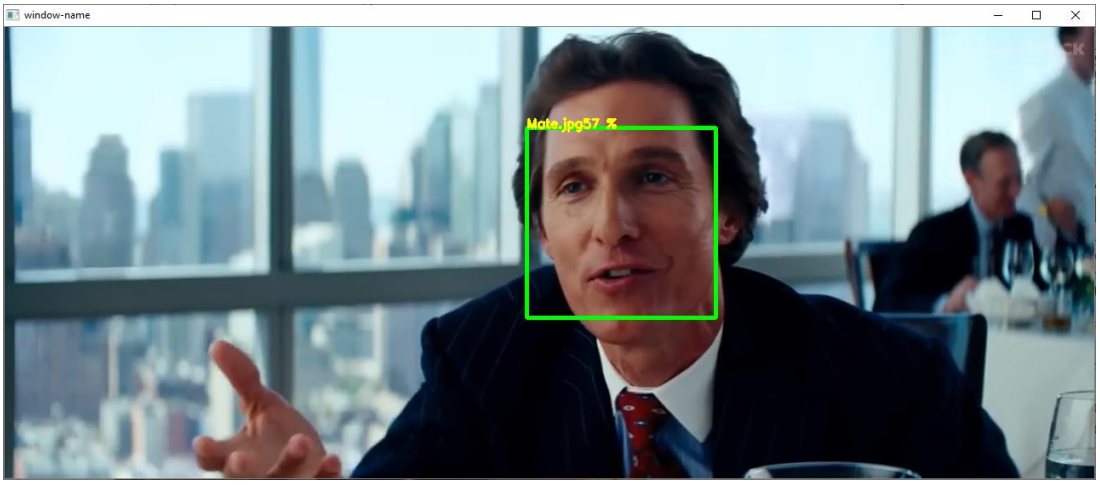

Рисунок 3.5 – пример работы алгоритма

Также стоит отметить что для идентификации по лицу, достаточно чтоб было видно большую часть лица, если на человеке будут надеты очки, то алгоритм сможет дать положительный результат, но, если на человеке надеты очки и появилась борода, велика вероятность низкой оценки схожести. Метки найден объектам ставятся по имени файла.

Алгоритм показывает хорошие результаты при выполнении поставленной задачи. К сожалению, из-за некоторых ограничении пришлось отказаться от работы распараллеливания на CUDA, но даже без этого, запуск алгоритма на новом железе и новом ПО возникает огромные проблемы. По этой причине требовалось создание виртуального пространства для запуска данного алгоритма, был создан контейнер DOCKER[24]. Также был создан сервер приложение использующие архитектуру REST-API, это позволяло перенести весь процесс выполнения алгоритма на сервер приложение, и сделать сам проект кроссплатформенным, так как сервер не зависим от типа клиента, который присылает HTTP запрос. Пример работы Java клиента показан на рисунках 3.6-3.7.

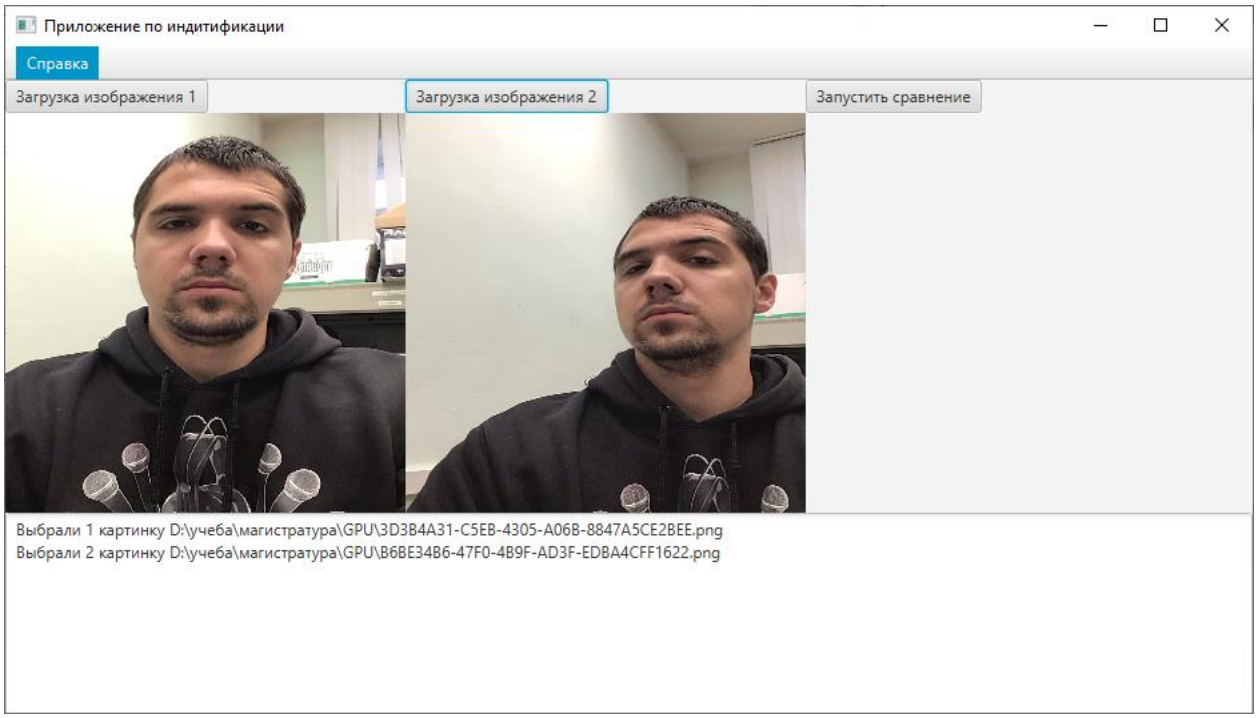

Рисунок 3.6 – пример работы клиента на java

Также был сделал в виде тестового режима клиент для telegram, vk и клиент для браузера, который использовал стандартные средства для html, где генерировался POST метод, и в ответ приходил Rest ответ, все эти 5варианты сделаны в тестовом режиме.

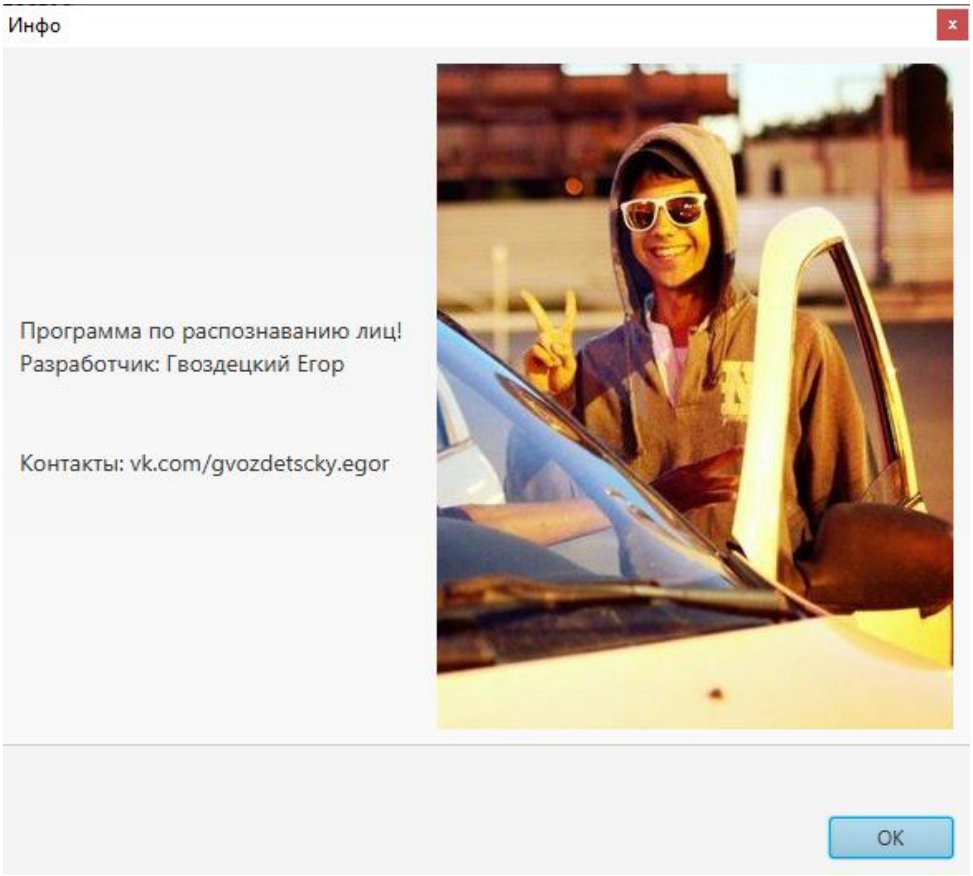

Рисунок 3.7 – Окно с информацией о программе

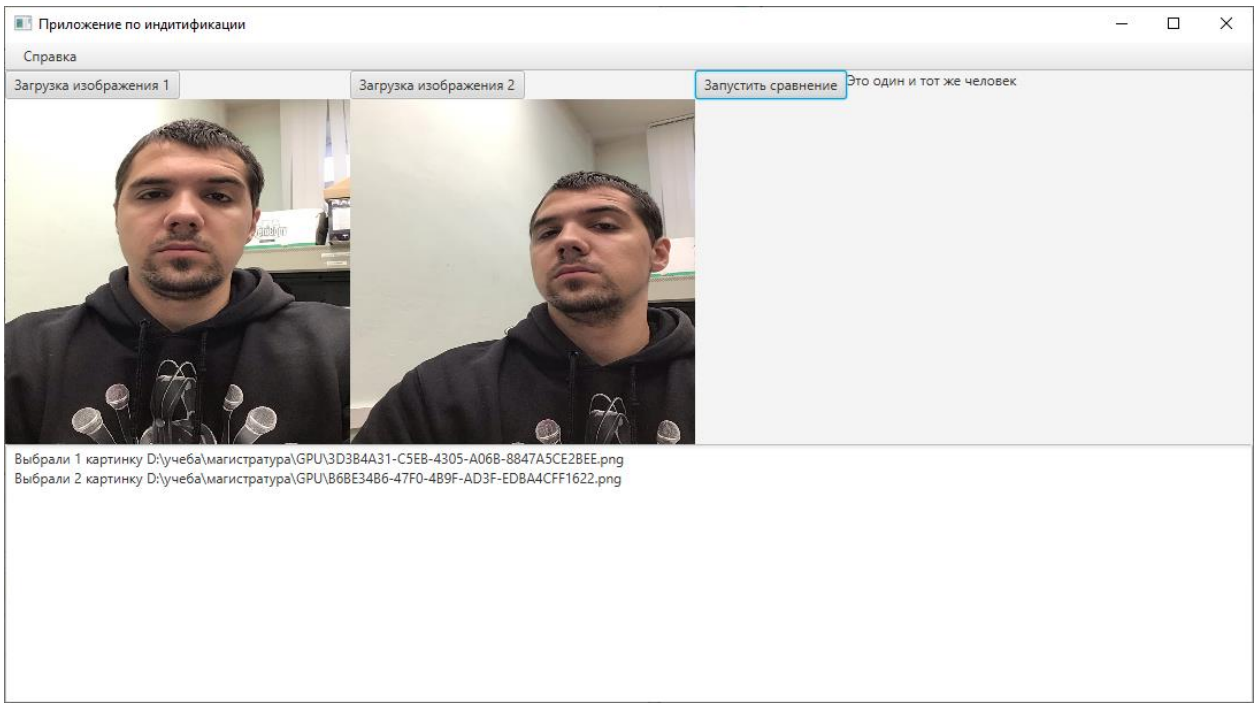

Рисунок 3.8 – пример работы клиента на java

Под итогом, было сделано 2 проекта, один по локализации объектов на изображении с применением алгоритмов глубокого обучения, сверточнные нейронные сети, модификации YOLO, для обеспеченья высокой производительности и адекватным результатом, также был сделан проект по идентификации лиц людей, для демонстрации применения локализации объектов. Во втором случае использование сверточнных нейронных сетей избыточно, лучше применить другие методы машинного обучения, они уменьшать скорость обработки, ухудшится качество идентификации, но такие алгоритмы подаются большей надстройки.

3.3 Реализация модуля идентификации человека по лицу

В разработанном приложении реализован функционал автоматизированной локализации лиц и идентификации человека по лицу.

Данный функционал клиента позволяет загружать фотографии на сервер, получать ответ и показывать его результаты, также присутствует окно логированния действий пользователя, и внедрено меню для дальнейшего развития программы, добавление возможности отправки на

сервер точность идентификации, новые возможности по определению пола человека и его настроения по лицу.

Алгоритм работы бизнес-процесса клиента:

1. Пользователь загружает по очереди две фотографии, если пользователь не загрузит их, то кнопка «Запустить сравнение» будет недоступна. Загружать можно файлы формата png/jpg.

2. После этого пользователь может нажать кнопку «Запустить сравнение» и фотографии будут переданы на сервер.

3. На клиент программу приходит ответ от сервера содержащий json, где есть errorMessage (Если был ответ с сервера с ошибкой, например, не прошла валидация, данное поле будет заполнено ошибкой сервера, иначе пустое) и поле data, содержащие признаки человека на фотографии, описание в главе 2, и значение определяющие идентификацию, значение от 0 до 1, чем ближе к 0, тем больше сходство, но само значение 0 является признаком, что фотографии одинаковые.

На стороне сервера был организовано взаимодействие с технологий CUDA, для получения наибольшей производительности в выполнении задачи локализации объектов на изображении с применением сверточнных нейронных сетей, что помогло увеличить производительность работы алгоритма в 10 раз по сравнение с вычислительной мощностью центральных процессоров вычислительной машины.

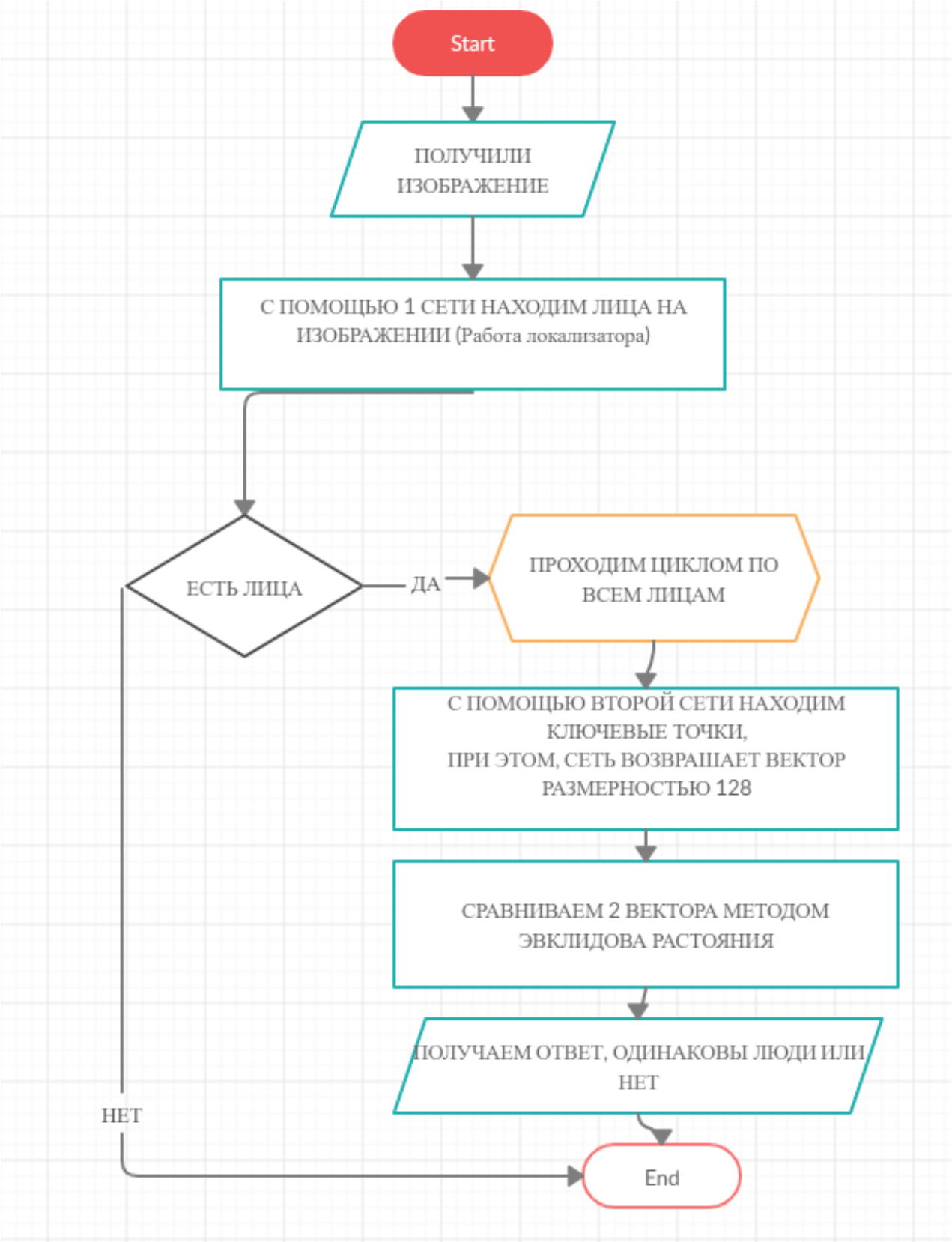

Рисунок 3.9 – Алгоритм работы идентификатора лиц

На рисунке 3.9 показан алгоритм идентификации человека. Предполагается, что программа уже знает вектор точек человека, которого должна идентифицировать.

1. Первым шагом мы должны получить изображение

2. Вторым шагом с помощью модуля локализации объектов с помощью сверточнной нейронной сети, который является темой исследования, обученной на нахождение лиц, которые направлены на камеру.

3.1 Если ни одного лица не найдено, то мы прерываем выполнение алгоритма

3.2 Если найдено хоть одно лицо, или множества, запоминаем их. Сохраняем данную информацию для легирования и анализа данных.

4 По найденным лицам поочередно запускаем алгоритм идентификации, который будет представлен дальше еще одной нейронной сетью.

5. С помощью второй нейронной сети, которая также является сверхточной, но выполняем другую задачу, сеть обучена находить контрольные точки, полный алгоритм представлен будет ниже

6 Полученный вектор мы сравнивает с вектором, который был получен до выполнения алгоритма, сравнение идет через функцию Эвклидова расстояния. Если значение функции является диапазон от 0 до заданного параметра сходимости, предполагаем, что человек на двух снимках является один и тем же. Если значение функции больше параметра сходимости, предполагаем, что это разные люди. При ситуации, если функция вернула 0, предполагаем, что это одинаковые фотографии.

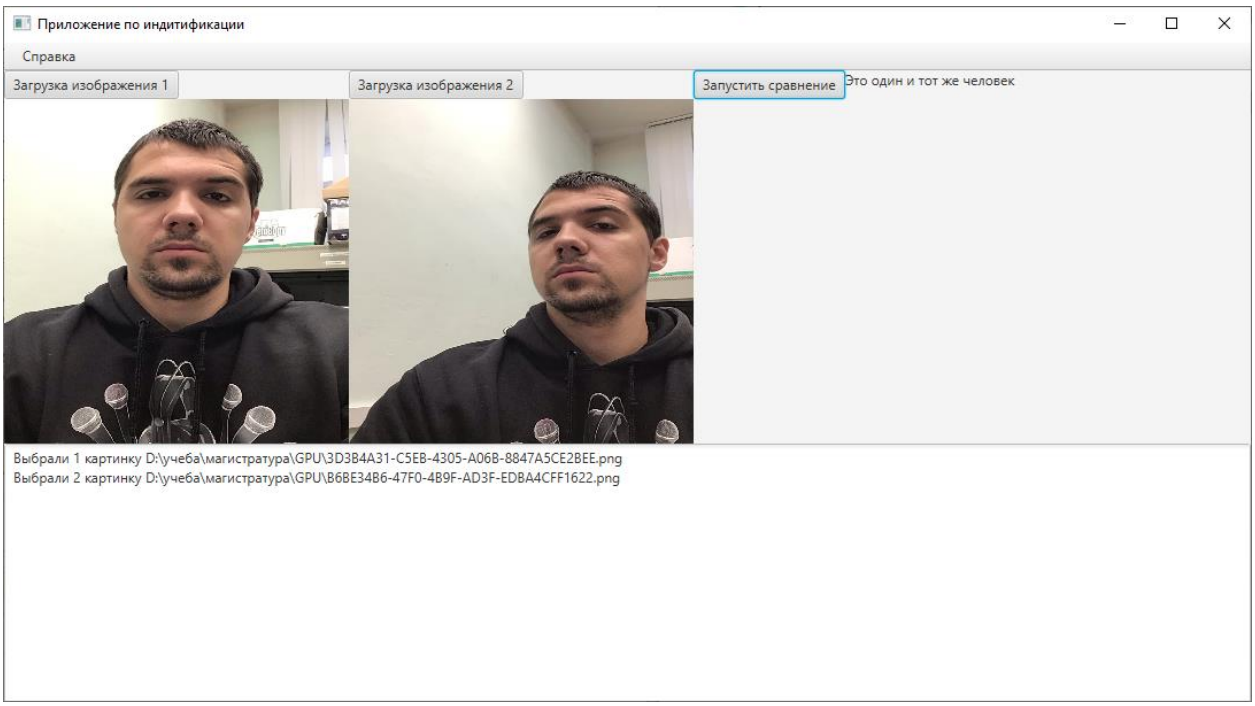

Рисунок 3.10 – Форма с результатами выполнения вычислительного эксперимента

В клиент-приложении присутствует дополнительная возможность экспорта данных вычислительного эксперимента в файл XML. Также в рамках научной исследовательской работы было проделана работа по реализации программного обеспечения применения сверточнные нейронные сети в задачи сегментации. Принцип работы похож на реализацию задачи локализации объектов, за исключением того, что нейронная сеть на выходном слое отображает изображение 3.10 которое является исходным изображением только размечает область по классам в виде цветов.

Рисунок 3.11 – Пример работы сверточнной нейронной сети в задачи сегментации

Как видно из исходного изображения 3.11 нейронная сеть смогла распределить 3 класса на изображении, машину, человека и пустое пространство, которое в этой задаче является обязательной пустой области при разметке. Практическое применение в машинном обучении у данной задачи является крайне значимым, так как решает задачу кластеризации объектов, и может применятся в задачах для выделения конкретных объектов на изображении 3.12.

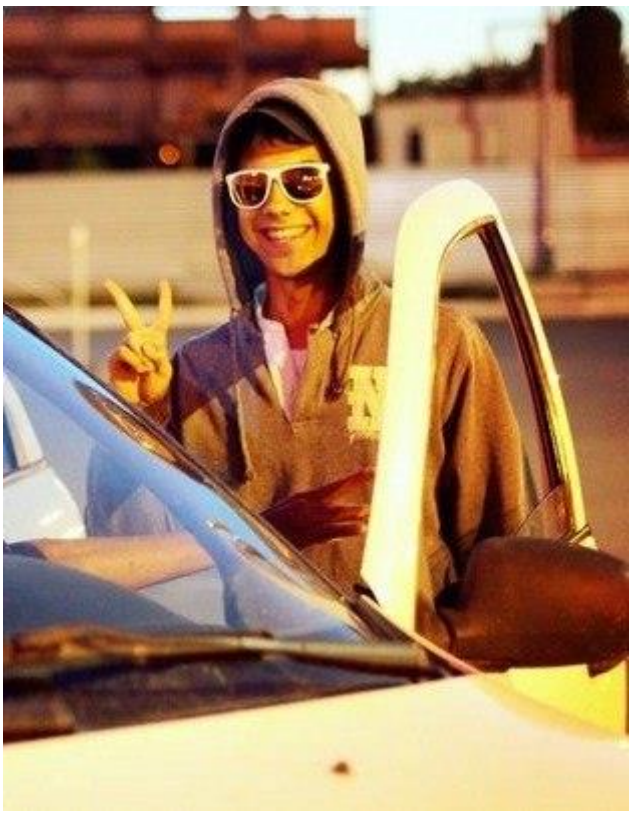

Рисунок 3.12 – Исходное изображение

Рисунок 3.13 – Пример работы сверточнной нейронной сети в задачи сегментации

Рисунок 3.14 – Исходное изображение

Рисунок 3.13 – Пример работы сверточнной нейронной сети в задачи сегментации

Сверточнная нейронная сеть сегментации тоже может решать задачу идентификации и может быть альтернативной нейронной сетью в работе локализации объектов. Но обладает следующими недостатками:

1. Низкая скорость работы, так как её задача не просто нахождение меток классов, а хранение полного изображения и выделение классов в работе алгоритма.

2. Низкая точность – выходное изображение может потерять много данных и для работы идентификации сильно сказаться на неправильные работы программы в целом.

В заключении было реализованы следующие программы, программа по локализации объектов в реальном времени с использованием сверточнных нейронных сетей, программа по идентификации людей по лицу с использованием 2 нейронных сетей и алгоритму Эвклидова расстояния и программа с использоавнием сверточнных нейронных сетей для решения задач сегментации.

Заключение

По результатам проведенных исследований были сделаны следующие выводы:

1. Одно из направлений совершенствования методов машинного обучения является повышение их степени автоматизации за счет снижение участие людей в анализе данных.

2. На данный момент в машинном зрении сверточнные нейронные сети могут решат очень сложные задачи для информационных технологий, но решать могут с ограничениями, для требуемой задачи должен присутствовать датасет достаточный и полный для получения обученной модели, количество классов не должно представлять из себя все объекты реального мира, а должно ограничивается определенной группой, на сегодняшний день это от 10 до 1000 классов, чем больше классов, тем менее пригодна для использования может получится модель нейронной сети.

3. На практической реализации показано, что создание своего датасета является плохой идеей, гораздо лучше использовать существующей датасет и докинуть в него требуемые данные, или до обучить уже готовую нейронную сеть, используя второй вариант и выбрав готовую, хорошо обученную нейронную сеть можно получить высокие результаты в задачи классификации и локализации.

4. Было проведено научное исследование использование сверточнных нейронных сетей для локализации объектов в реальном времени, используя вычислительные мощности центрального процессора и графического процессора. Результаты исследования показали огромный рост производительности алгоритма с использованием графических процессоров в среднем в 10 раз.

5. Проведено реализация сверточнной нейронной сети в задачи сегментации и представлены результаты показывающие её практическое применение в машинном зрении на примере задачи идентификации и автоматической задачи выделении объекта на изображении.

6. Экспериментально показана значимость сравнений векторов с использованием Эвклидова расстояния для решения задачи идентификации лица для решения автоматической задачи.

7. Разработано приложение практического отображения задачи локализации объектов в реальном времени с использованием сверточнной нейронной сети и представлены результаты точности и использование ускорения работы алгоритма с использованием графических процессоров.

8. Научная новизна исследования – доказано что скорость работы алгоритма может быть увеличена в разы с использованием сверточнных нейронных сетей и графических процессоров, а также доказана что задача локализации является более точной, чем задача сегментации, хотя практически они решают похожие проблемы машинного зрения.

9. Практическая значимость работы заключается в разработке программного обеспечения с использованием сверточнных нейронных сетей для задачи локализации объектов на изображении и создание программного продукта для использования практического опыта в решении более сложных задача, например, в решении задачи идентификации лиц людей, выделения классов на изображении или выявлений признаков у этих объектов, как настроение у людей, жесты.

Список используемых источников

1. Fast image scanning with deep max-pooling convolutional neural networks. [Текст] / A. Giusti, D. C. Cires J. Masci, L. M. Gambardella, and J. Schmidhuber. — In ICIP, 2013.
2. Harley, A. W. “An Interactive Node-Link Visualization of Convolutional Neural Networks [Текст] — in ISVC, 2015
3. Hariharan B. Semantic contours from inverse detectors. [Текст] / P. Arbelaez, L. Bourdev, S. Maji, and J. Malik — In International Conference on Computer Vision (ICCV), 2011.
4. B. Hariharan, taneous detection and segmentation. [Текст] / P. Arbelaez, R. Girshick, J. Malik. Simul — In European Conference on Computer Vision (ECCV), 2014.
5. B. Hariharan, Hypercolumns for object segmentation and fine-grained localization [Текст] ./ P. Arbelaez, R. Girshick, J. Malik. — In Computer Vision and Pattern Recognition, 2015.
6. He K., Spatial pyramid pooling in deep convolutional networks for visual recognition. [Текст] / X. Zhang, S. Ren, and J. Sun. — In ECCV, 2014.
7. Caffe: Convolutional architecture for fast feature embedding. [Текст] / Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. — arXiv preprint arXiv:1408.5093, 2014.
8. Backpropagation applied to hand-written zip code recognition. [Текст] / Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. — In Neural Computation, — 1989.
9. Backpropagation applied to hand-written zip code recognition. [Текст] / Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. — In Neural Computation, 1989.
10. LeCun, Y. A. Efficient backprop. In Neural networks: Tricks of the trade. [Текст] / L. Bottou, G. B. Orr, and K.-R. Muller. — Springer, 1998.

11. Liu, C. Sift flow: Dense correspondence across scenes and its applications. Pattern Analysis and Machine Intelligence [Текст] / J. Yuen, and A. Torralba — IEEE Transactions on, 33(5):978–994, 2011.

12. Long J., Do convnets learn correspondence? [Текст] / N. Zhang, and T. Darrell. — In NIPS, 2014.

13. Long, J. Shelhamer, E. Darrell, T. Fully Convolutional Networks for Semantic Segmentation [Текст]/ UC Berkeley Chen, Z. Real-time transient stability status prediction using cost-sensitive extreme learning machine / Zhen Chen, Xianyong Xiao, Changsong Li, Yin Zhang, Qingquan Hu // Neural Computing and Applications. – 2016. – №27. – pp. 330-333

14. Blum, A. Special Issue on New Theoretical Challenges in Machine Learning / Avrim Blum, Philip M. Long // Algorithmica. – 2015. – №72. – pp. 191-192

15. Bing, L. Lifelong machine learning: a paradigm for continuous learning / Bing Liu // Machine learning. – 2016. – №3. – pp. 1-3.

16. Günnemann, S. Machine Learning Meets Databases / Stephan Günnemann // Datenbank-Spektrum. – 2017. – №17. – pp. 77-83

17. Langley, P. Research papers in machine learning / Pat Langley // Machine Learning. – 1987. – №2. – pp. 195-198

18. Abdolrazzaghi, M. Fast-forward solver for inhomogeneous media using machine learning methods: artificial neural network, support vector machine and fuzzy logic / Mohammad Abdolrazzaghi, Soheil Hashemy, Ali Abdolali // Neural Computing and Applications. – 2016. – №3. – pp. 1-9.

19. Carbonell, J. Machine Learning: A maturing field / Jaime Carbonell // Machine Learning. – 1992. – №9. – pp. 5-7.

20. Shen, Q. Decay-weighted extreme learning machine for balance and optimization learning / Qing Shen, Xiaojuan Ban, Ruoyi Liu, Yu Wang // Machine Vision and Applications. – 2017. – №3. – pp. 1-11.

21. Shen, Q. Decay-weighted extreme learning machine for balance and optimization learning / Qing Shen, Xiaojuan Ban, Ruoyi Liu, Yu Wang // Machine Vision and Applications. – 2017. – №3. – pp. 1-11.

22. Martin T Hagan, Howard B Demuth, Mark H Beale, Orlando De Jesús. Neural Network Design (2nd edition) // Martin Hagan; 2 edition. -2014. -pp. 150-200.

23. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning (Adaptive Computation and Machine Learning series) // The MIT Press. -2016. -pp. 700-710.

24. Raúl Rojas. Neural Networks: A Systematic Introduction // Springer Berlin Heidelberg; 1 edition. -1996. -pp. 111-150.

25. Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics) // Springer. -2006. -pp. 638-700.

26. Toby Segaran. Programming Collective Intelligence: Building Smart Web 2.0 Applications. // Shroff; First edition. -2011. -pp. 250-350.

27. Tariq Rashid. Make Your Own Neural Network // CreateSpace Independent Publishing Platform; 1 edition. -2016. -pp. 50-70.

28. Sebastian Raschka. Python Machine Learning, 1st Edition // Packt Publishing. -2015. -pp. 385-428.

29. Yaser S. Abu-Mostafa. Learning from data // AMLBook. -2012. -pp. 45-200.

30. Stuart Russell. Artificial Intelligence: Pearson New International Edition: A Modern Approach // Pearson; 3 edition. -2013. -pp. 800-1000.

31. Simon S Haykin. Neural Networks and Learning Machines (3rd Edition) // PHIL. -2010. -pp. 555-610

32. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems // O'Reilly Media; 1 edition. -2017. -pp. 320-405.

33. Sandhya Samarasinghe. Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition // Auerbach Publications; 1 edition. -2006. -pp. 200-300.

34. Amit Konar. Emotion Recognition: A Pattern Analysis Approach // Wiley; 1 edition. -2015. -pp. 460-550.

35. François Chollet. Deep Learning with Python // Manning Publications; 1st edition. -2017. -pp. 113-180.

36. Richard S. Sutton. Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series) // A Bradford Book; second edition. -2018. -pp. 350-370.

37. Anirudh Koul. Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & TensorFlow // O'Reilly Media; 1 edition. -2019. -pp. 570-600.

38. Andrew Trask. Grokking Deep Learning // Manning Publications; 1st edition. -2019. -pp. 115-200.

39. Andriy Burkov. The Hundred-Page Machine Learning Book // Andriy Burkov. -2019. -pp. 10-100.

40. Sandro Skansi. Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence (Undergraduate Topics in Computer Science) // Springer; 1st edition. -2018. -pp. 25-90.

41. Christoph Molnar. Interpretable Machine Learning // lulu.com. -2020. -pp. 150-200.

42. E. R. Davies. Computer and Machine Vision: Theory, Algorithms, Practicalities // Academic Press; 4 edition. -2012. -pp. 700-900.

43. Simon J. D. Prince. Computer Vision: Models, Learning, and Inference // Cambridge University Press; 1 edition. -2012. -pp. 404-500.

44. Shai Shalev-Shwartz. Understanding Machine Learning: From Theory to Algorithms // Cambridge University Press; 1 edition. -2014. -pp. 350-400.

45. Joshua Bloch. Effective Java // Addison-Wesley Professional; 3 edition. -2018. -pp. 201-250.

46. Herbert Schildt. Java: The Complete Reference, Eleventh Edition // McGraw-Hill Education; 11 edition. -2018. -pp. 800-1100.

47. Cay S. Horstmann. Core Java Volume I--Fundamentals // Prentice Hall; 11 edition. -2018. -pp. 666-750.

48. Benjamin J. Evans. Java in a Nutshell: A Desktop Quick Reference // O'Reilly Media; 7 edition. -2019. -pp. 150-255.

49. Joel Murach. Murach's Java Programming // Mike Murach & Associates; 5 edition. -2017. -pp. 490-530.

50. Raoul-Gabriel Urma. Modern Java in Action: Lambdas, streams, functional and reactive programming // Manning Publications; 2nd edition. -2018. -pp. 340-400.

51. Julie Anderson. Java Illuminated // Jones & Bartlett Learning; 5 edition. -2018. -pp. 943-1150.

52. Гвоздецкий, Е.А. Система компьютерного зрения для локализации объектов на изображениях / Е.А. Гвоздецкий [и др.] // сборник статей всероссийской научно-практической междисциплинарной конференции «Молодежь. Наука. Общество» – Тольятти: – 2018. - №7 (31) – с. 119-122.

53. Гвоздецкий, Е.А. Использование свертоных нейронных сетей для локализации объектов на изображении / Е.А. Гвоздецкий [и др.] // сборник статей всероссийской научно-практической междисциплинарной конференции «Молодежь. Наука. Общество» – Тольятти – 2018. – с. 122-127.

54. Гвоздецкий, Е.А. СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМА K-MEANS / Е.А. Гвоздецкий [и др.] // сборник статей V Международной научно-практической конференции (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук» – Тольятти – 2019. – с. 16-19.

55. Гвоздецкий, Е.А. КЛАСТЕРИЗАЦИЯ ИЗОБРАЖЕНИЯ НА ОСНОВЕ ОЦЕНКИ ГЕОМЕТРИЧЕСКОЙ ФОРМЫ ОБЪЕКТОВ / Е.А.

Гвоздецкий [и др.] // сборник статей V Международной научно-практической конференции (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук» – Тольятти – 2019. – с. 502-506.

56. технических наук» – Тольятти – 2019. – с. 502-506.