Министерство образования и науки Республики Казахстан
НАО «Алматинский университет энергетики и связи»
Кафедра «Электроника и робототехника»
Специальность 5В071600 – «Приборостроение»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
к. т. н., доцент
_____ Чигамбаев Т. О.
«____» _____2020 г.

**ДИПЛОМНЫЙ ПРОЕКТ**

На тему: «Разработка интеллектуальной системы управления технологического процесса сортировки деталей на базе микрокомпьютера Raspberry Pi – 3»

Специальность 5В071600 - Приборостроение
Выполнила Дуйсенбек Инкар Дауренкызы Группа ПС(КИП)-16-4
(Ф.И.О.)
Научный руководитель доц. Байкенов Бахытжан Сергеевич
(ученая степень, звание, Ф.И.О.)
Консультанты:
по экономической части: _____
(ученая степень, звание, Ф.И.О.)
_____ «___» _____201___г.
по безопасности жизнедеятельности: _____
(ученая степень, звание, Ф.И.О.)
_____ «___» _____201___г.
Нормоконтролер:_____
(ученая степень, звание, Ф.И.О.)
_____ «____» _____201___г.
Рецензент: _____
(ученая степень, звание, Ф.И.О.)
_____ «____» _____201___г.

Алматы 2020

Министерство образования и науки Республики Казахстан
НАО «Алматинский университет энергетики и связи»
Институт «Космическая инженерия и телекоммуникации»
Кафедра «Электроника и робототехника»
Специальность 5В071600 – «Приборостроение»

**ЗАДАНИЕ**

на выполнение дипломной работы студента
Дуйсенбек Инкар Дауренкызы

Тема работы: «Разработка интеллектуальной системы управления технологического процесса сортировки деталей на базе микрокомпьютера Raspberry Pi – 3».

Утверждена приказом ректора № от «» октября 2019 г.

Срок сдачи законченной работы «20» мая 2020 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): система распознавания объектов

- стационарная установка;
- локальное хранилище данных;
- микрокомпьютер Raspberry Pi - 3;
- температура среды от +10 до +30 градусов Цельсия;
- колебаний и вибраций поверхности не предусмотрено.

Перечень вопросов, подлежащих разработке в дипломной работе, или краткое содержание дипломной работы:

- анализ и исследование интеллектуальных систем;
- изучение и описание принципа машинного обучения;
- сбор и анализ базы данных объектов;
- разработка интеллектуальной системы распознавания и сортировки объектов;
- глубокое обучение интеллектуальной системы;
- рассмотрение вопросов охраны труда и БЖД обслуживающего персонала;
- расчет технико-экономической эффективности проекта.

Перечень графического материала (с точным указанием обязательных чертежей):

Основная рекомендуемая литература:

Консультации по работе (проекту) с указанием относящихся к ним разделов работы (проекта)

| Раздел | Консультант | Сроки | Подпись |
|---|---|---|---|
| Технологическая часть | Байкенов Б.С. | | |
| Конструкторская часть | Байкенов Б.С. | | |
| Программное обеспечение | Байкенов Б.С. | | |
| Экономическая часть | Бекишева А.И. | | |
| Охрана труда и БЖД | Приходько Н.Г. | | |

График
выполнения дипломной работы

| Наименование разделов, перечень разрабатываемых вопросов | Сроки представления научному руководителю | Приме чание |
|---|---|---|
| Введение | | |
| 1. Технологическая часть | | |
| 1.1 Изучение искусственного интеллекта, машинного и глубокого обучения | | |
| 1.2 Математические основы нейронных сетей | | |
| 1.3 Выбор системы распознавания | | |
| 2. Конструкторская часть | | |
| 2.1 Выбор микрокомпьютера | | |
| 2.2 Внешние устройства системы управления | | |
| 2.3 Схема системы управления | | |
| 3. Программное обеспечение | | |
| 3.1 Блок-схема программы распознавания | | |
| 3.2 Листинг программы распознавания | | |
| 4. Охрана труда и БЖД | | |
| 5. Экономическая часть | | |
| Оформление работы | | |

Дата выдачи задания «5» января 2020 г.

Зав. кафедрой ЭиР Чигамбаев Т.О.
Научный руководитель Байкенов Б.С.
Задание приняла Дуйсенбек И.Д.

## Аңдатпа

Бұл дипломдық жоба өндірісте бөлшектерді сұрыптаудың технологиялық процесін басқарудың зияткерлік жүйесін құруға арналған. Тану және сұрыптау жүйелерін талдау негізінде кіріс деректерін іріктеу және өңдеу алгоритмін таңдау, сенсорлық бөліктердің мәліметтер базасын құру, тірек құрылымын құру және бағдарламалық жасақтаманы жасау жүзеге асырылды. Жоба прототип ретінде жүзеге асырылды, оның техникалық шешімі техникалық-экономикалық негіздемесімен расталды.

## Аннотация

Данный дипломный проект посвящен разработке интеллектуальной системы управления технологического процесса сортировки деталей на производстве. На основе анализа систем распознания и сортировки осуществлен выбор алгоритма выборки и обработки входных данных, создание базы данных деталей датчиков, разработка несущей конструкции и создание программного обеспечения. Проект выполнен в виде прототипа, техническое решение которого подтверждено технико-экономическим обоснованием.

## Annotation

This graduation project is dedicated to the development of an intelligent control system for the technological process of sorting parts in production. Based on the analysis of recognition and sorting systems, the selection of an algorithm for sampling and processing input data, the creation of a database of sensor parts, the development of the supporting structure, and the creation of software were performed. The project was implemented as a prototype, the technical solution of which was confirmed by a feasibility study.

## СОДЕРЖАНИЕ


Введение 6
1 Технологическая часть 7
1.1 Постановка задачи 7
1.2 Изучение интеллектуальных систем глубокого обучения 15
1.3 Принцип работы и основы нейронных сетей 20
1.4 Глубокое обучение в технологиях компьютерного зрения 31
2 Конструкторская часть 41
2.1 Микрокомпьютер как основа системы 41
2.2 Камера как основной элемент считывания данных 50
2.5 Схема управления системы 52
3 Программное обеспечение 55
3.1 Среда разработки и листинг нейросети 55
3.3 Блок схема и программа для распознавания 59
4 Безопасность жизнедеятельности 62
4.1 Основание актуальности дипломного проекта 62
4.2 Обоснование дипломного проекта 62
4.3 Улучшения в сфере безопасности 63
4.4 Обоснование темы дипломного проекта 63
4.5 Требования к рабочему месту 67
4.6 Режим труда 69
4.7 Расчет освещенности 70
4.8 Расчет уровня шума 73
5 Технико-экономическое обоснование 76
5.1 Основание актуальности дипломного проекта 76
5.2 Обоснование дипломного проекта 76
5.3 Экономические показатели 77
5.4 Обоснование темы дипломного проекта 77
5.5 Расчет затрат на разработку продукта 80
5.6 Экономический эффект и эффективность 84
Заключение 87
Список литературы 88
Приложение А 89

## Введение

На сегодняшний день существует огромные перспективы в области изучения и разработки интеллектуальных систем машинного обучения. Даже не за десятилетие отрасль искусственного интеллекта (ИИ) претерпела небывалое изменение, начиная от распознавания образов и речи – которое оставляло желать лучшего, до эффективного выполнения этих задач.

Такой стремительный прогресс неминуемо отразился на многих отраслях, естественно включая промышленность. Однако, процесс внедрения новых технологий всегда граничит с внушительным количеством проблем, поэтому чтобы начать процесс насаждения технологий машинного обучения, для начала выполнения задач, которые можно решить при её применении. А именно, технология должна быть доступна как можно большему пласту людей, не являющихся специалистами. Здесь уже можно вывод о том, чтобы максимально воспользоваться всем потенциалом машинного обучения, нужно популяризировать и упрощать их так, чтобы каждый мог внести вклад в развитие интеллектуальных систем.

Искусственный интеллект используется во многих аспектах производства для увеличения срока исправного функционирования технических средств и преумножения времени их бесперебойной работы. Возможности использования машинного и обучения систем стали применяться в областях робототехники и искусственного зрения в задаче развития технологий промышленных роботов. В тоже время, следует отметить, что данная отрасль мало проработана на территории Казахстана, как и в странах СНГ в целом. Причиной этому, является малое по своему охвату использование роботов на производствах.

Исходя из вышесказанного, следует вывод, что машинное обучение является мощным инструментом для постепенной автоматизации ИИ. С незапамятных времен люди создавали лучшие и лучшие инструменты для понимания и контроля окружающей среды вокруг нас. И именно глубокое обучение — это сегодняшняя глава в истории инноваций. Это особенно необычно, учитывая, что глубокое обучение, похоже, использует в основном тот же алгоритм (нейронные сети) для достижения обширного количества инструментов. Хотя глубокое обучение все еще активно развивается, недавние открытия наводят нас на мысли, что на самом деле мы обнаружили больше, чем просто отличный инструмент, но и окно в наши собственные умы.

Определение машинного обучения можно сформировать так: машинное обучение представляет собой науку (и искусство) программирования компьютеров для того, чтобы они могли обучаться на основе данных.

# 1 Технологическая часть

## 1.1 Постановка задачи

Темой данного дипломного проекта, является: «Разработка интеллектуальной системы управления технологического процесса сортировки деталей на базе микрокомпьютера Raspberry Pi – 3», и исходя из этого, можно точно обозначить цель проекта:

Исследовать и разработать интеллектуальную систему, методом глубокого обучения, для внедрения в технологический процесс сортировки деталей, на базе 4-ядерного одноплатного микрокомпьютера Raspberry Pi, модели 3B+.

Далее, как само собой разумеющееся, следует постановка задач, которые должны быть исследованы и проанализированы в ходе данного проекта:

1) Изучить теоретические аспекты разработки интеллектуальных систем глубокого обучения;

2) Проанализировать математическую модель архитектуры нейронной сети;

3) Сравнить аппаратное обеспечение различных микрокомпьютеров;

4) Разработать нейронную сеть распознавания объектов.

Система должна значительно упрощать работу узла сортировки, и давать возможность оператору легче управлять технологическим процессом. Также в результате дальнейших исследований, появится возможность утверждать, что внедрение системы будет вносить весомый вклад в экономию денежных средств, а также не иметь пагубного влияния на условия жизнедеятельности человеческих ресурсов.

## 1.2 Изучение интеллектуальных систем глубокого обучения

Глубокое обучение (Deep Learning) — это подмножество машинного обучения, которое является областью, посвященной изучению и разработке машин, которые могут обучаться (иногда с целью в конечном итоге достичь общего искусственного интеллекта). В промышленности глубокое обучение используется для решения практических задач в различных областях, таких как компьютерное зрение (изображение), обработка естественного языка (текст) и автоматическое распознавание речи (аудио). Можно сказать, глубокое обучение — это подмножество методов в наборе инструментов машинного обучения, в первую очередь использующих искусственные нейронные сети, которые представляют собой класс алгоритмов, отдаленно вдохновленных человеческим мозгом.

Следует обратить внимание на рисунок 1.1, и понять, что не все глубокое обучение сосредоточено на использовании обобщенного искусственного интеллекта. На самом деле, многие применения этой технологии применяются для решения широкого круга задач в промышленности. Поэтому, следует стремится сосредоточиться на изучении

основ глубокого обучения, лежащих в основе передовых исследований и промышленности.


Рисунок 1.1 – Изображение пересекающихся множеств ИИ и методов его обучения

«Область обучения, которая дает компьютерам возможность учиться без явного программирования», - так описал о машинном обучении А. Самюэль.

Учитывая, что глубокое обучение является подмножеством машинного обучения, нужно дать определение что такое машинное обучение. В общем, это то, что подразумевает его название. Машинное обучение — это область компьютерных наук, в которой машины учатся выполнять задачи, для которых они не были запрограммированы явно. Короче говоря, машины наблюдают шаблон и пытаются имитировать его каким-либо образом, который может быть прямым или косвенным (рисунок 1.2).

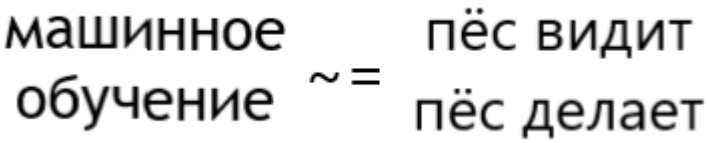


Рисунок 1.2 – Простое сравнение принципа машинного обучения

Тут стоит упомянуть, прямое и косвенное подражание как параллель двум основным типам машинного обучения: контролируемое машинное обучение и неконтролируемое машинное обучение (рисунок 1.3). Контролируемое машинное обучение — это прямая имитация схемы между двумя наборами данных. Он всегда пытается взять входной набор данных и преобразовать его в выходной набор данных. Это может быть невероятно мощной и полезной возможностью. Рассмотрим следующие примеры:

а) использование пикселей изображения для обнаружения присутствия или отсутствия кошки;

б) использование фильмов, которые вам понравились, чтобы предсказать фильмы, которые вам могут понравиться;

в) использование чьих-либо слов, чтобы предсказать, счастливы они или грустны;

г) использование данных датчика погоды для прогнозирования вероятности дождя;

д) использование датчиков автомобильного двигателя для прогнозирования оптимальных настроек тюнинга;

е) использование новостных данных для прогнозирования цены акций завтрашнего дня;

ж) использование входного числа для прогнозирования числа вдвое больше его размера;

з) использование необработанного аудиофайла для прогнозирования стенограммы аудио.

Все эти задачи контролируются машинным обучением. Во всех случаях алгоритм машинного обучения пытается имитировать шаблон между двумя наборами данных таким образом, чтобы он мог использовать один набор данных для прогнозирования другого. Для любого приведенного выше примера представьте, что у вас была возможность предсказать выходной набор данных, исходя только из входного набора данных. Это было бы глубоко.

Контролируемое машинное обучение. Контролируемое обучение — это метод преобразования одного набора данных в другой. Например, если у нас был набор данных «Цены на приборы в понедельник», в котором записывались цены на каждый прибор в каждый понедельник за последние 10 лет, и второй набор данных «Цены на приборы во вторник», записанные за тот же период времени, это контролируемое обучение - алгоритм может попытаться использовать один для прогнозирования другого.


Рисунок 1.3 – Пример прогнозирования с контролируемым обучением

Следует, что если мы успешно обучаем наш контролируемый алгоритм машинного обучения в течение 10 лет по понедельникам и вторникам, то мы можем предсказать цену приборов в любой вторник в будущем, учитывая цену приборов в непосредственно предшествующий понедельник. Тут стоит остановиться и обдумать это на мгновение.

Управляемое машинное обучение — это хлеб прикладного искусственного интеллекта (то есть «Узкий ИИ»). Это полезно для того, чтобы взять то, что мы знаем, как ввод и быстро преобразовать это в то, что мы хотим знать. Это позволяет контролируемым алгоритмам машинного обучения расширять человеческий интеллект и возможности, казалось бы, бесконечным количеством способов (рисунок 1.4)

Большая часть работы, использующей машинное обучение, приводит к обучению некоторого контролируемого классификатора. Даже неконтролируемое машинное обучение (который описан далее) обычно делается для того, чтобы помочь в разработке точного контролируемого алгоритма машинного обучения.


Рисунок 1.4 – Алгоритм работы контролируемого обучения

Неконтролируемое машинное обучение. Обучение без контроля группирует ваши данные.

Неконтролируемое обучение имеет свойство, общее с контролируемым обучением. Он превращает один набор данных в другой. Однако набор данных, в который он преобразуется, ранее не был известен или понят. В отличие от контролируемого обучения, не существует «правильного ответа», который мы пытаемся заставить модель дублировать. Мы просто сообщаем неконтролируемому алгоритму «найти шаблоны в этих данных и рассказать мне о них».

Например, кластеризация набора данных в группы является типом обучения без учителя. «Кластеризация» преобразует вашу последовательность точек данных в последовательность меток кластера. Если он изучает 10 кластеров, то эти метки обычно являются числами 1-10. Каждой точке данных будет присвоен номер, в зависимости от того, в каком кластере он находится. Таким образом, ваш набор данных превращается из группы точек данных в группу меток. (рисунок 1.5).

Рисунок 1.5 – Алгоритм работы неконтролируемого обучения

Параметрическое и непараметрическое обучение. Чрезмерно упрощенное: обучение методом проб и ошибок в сравнении со счетом и вероятностью

Последние две страницы разделили все наши алгоритмы машинного обучения на две группы, контролируемые и не контролируемые. Теперь мы собираемся обсудить другой способ разделения одних и тех же алгоритмов машинного обучения на две группы: параметрическую и непараметрическую. Итак, если мы подумаем о нашем маленьком облаке машинного обучения, у него есть две настройки (рисунок 1.6)

Как видите, у нас действительно есть четыре различных типа алгоритмов на выбор. Алгоритм либо не контролируется, либо контролируется, и он либо параметрический, либо непараметрический. В то время как предыдущий раздел о надзоре действительно касался типа изучаемого паттерна, параметризм - о том, как хранится обучение, а зачастую и о методе обучения. Во-первых, давайте посмотрим на формальное определение параметризма против непараметризма. Для протокола, есть еще некоторые споры о точной разнице.

Параметрическая модель характеризуется наличием фиксированного числа параметров, тогда как количество параметров непараметрической модели является бесконечным (определяется данными).


Рисунок 1.6 – Четыре вида алгоритмов обучения

Контролируемое параметрическое обучение. Упрощенное: обучение методом проб и ошибок с помощью тумблеров.

Контролируемые параметрические обучающие машины — это машины с фиксированным числом ручек (это параметрическая часть), причем обучение происходит путем поворота ручек. Входные данные поступают, обрабатываются в зависимости от угла поворота и преобразуются в прогноз (рисунок 1.7).

Обучение осуществляется путем поворота ручек под разными углами. Если мы пытаемся предсказать вероятность того, что Red Socks выиграет Мировую Серию, то эта модель сначала будет брать данные (такие как спортивная статистика, такая как победа / поражение или среднее число пальцев) и делать прогноз (такой как 98 % шанс) Затем модель будет наблюдать, действительно ли победили Красные носки. После того, как он узнал, победили ли они, наш алгоритм обучения обновит ручки, чтобы сделать более точный прогноз, когда в следующий раз увидит те же / похожие входные данные.

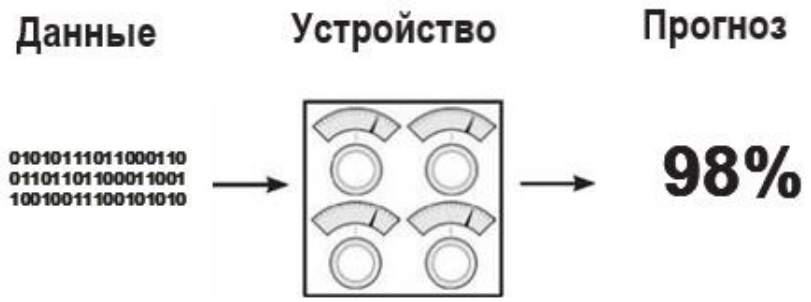


Рисунок 1.7 – Схематичное представление параметрической обучающей машины

Возможно, он «повернул бы» ручку «победа / поражение», если бы победа / поражение команды было хорошим предиктором. И наоборот, он мог бы выключить ручку «среднее число пальцев», если бы эта точка данных не была хорошим предиктором. Так учатся параметрические модели!

Обратите внимание, что все то, что изучила модель, может быть зафиксировано в положениях ручек в любой момент времени. Можно также думать об этом типе модели обучения как об алгоритме поиска. Мы «ищем» подходящую конфигурацию регулятора, пытаясь настроить его, настроить и повторить попытку.

Далее отметим, что понятие проб и ошибок не является формальным определением, но оно является очень распространенным (за исключением) свойством параметрических моделей. Когда есть произвольное (но фиксированное) число нобов, которые нужно повернуть, тогда требуется некоторый уровень поиска, чтобы найти оптимальную конфигурацию. Это в отличие от непараметрического обучения, которое часто основано на подсчете и (более или менее) «добавляет новые ручки», когда оно находит что-то новое для подсчета. Давайте разберем контролируемое параметрическое обучение на три этапа.

Шаг 1: предсказание.

Чтобы проиллюстрировать контролируемое параметрическое обучение, давайте продолжим нашу спортивную аналогию, где мы пытаемся предсказать, выиграют ли Red Socks Мировую Серию. Первым шагом, как уже упоминалось, является сбор спортивной статистики, отправка ее через

нашу машину и прогнозирование вероятности победы Red Socks (рисунок 1.8).

Шаг 2: сравнение с истиной.

Второй шаг - сравнить прогноз (98%) с моделью, которая нас интересует (победили ли красные носки). К сожалению, они проиграли, поэтому наше сравнение таково (рисунок 1.9).

Этот шаг просто признает, что, если бы наша модель прогнозировала 0%, она бы отлично предсказала предстоящую потерю команды. Мы хотим, чтобы наша машина была точной, что приводит нас к шагу 3.


Рисунок 1.8 – Прогноз вероятности победы

Pred: **98% >** Truth: **0%**

Рисунок 1.9 – Сравнение прогноза

Шаг 3: изучение образца.

Этот шаг регулирует тумблеры, изучая как много пропущено моделью (98%), и каковы были входные данные (спортивная статистика) во время прогнозирования. Затем он поворачивает ручки, чтобы сделать более точный прогноз с учетом входных данных. Теоретически, в следующий раз, когда он увидит ту же спортивную статистику, прогноз будет ниже 98%. Обратите внимание, что каждая ручка представляет чувствительность предсказания к различным типам входных данных. Это то, что мы меняем, когда мы «учимся».

Неконтролируемое параметрическое обучение

Параметрическое обучение без учителя использует очень похожий подход. Давайте пройдемся по ступенькам на высоком уровне. Помните, что неконтролируемое обучение — это группировка ваших данных. Параметрическое обучение без учителя использует ручки для группировки ваших данных. Однако в этом случае обычно для каждой группы имеется несколько ручек, каждая из которых отображает привязку ваших входных данных к этой конкретной группе (за исключением и нюансом — это описание высокого уровня). Давайте рассмотрим пример, в котором мы предполагаем, что хотим разделить наши данные на три группы (рисунок 1.10).

В наборе данных на рисунке 10 мы идентифицировали три кластера в данных, которые мы могли бы хотеть найти в нашей параметрической модели. Я идентифицировал их с помощью форматирования как группа 1, группа 2 и группа 3. Давайте расскажем о нашем первом назначении данных с помощью обученной неконтролируемой модели ниже. Обратите внимание, что он наиболее сильно соответствует группе один.

Машина каждой группы пытается преобразовать входные данные в число от 0 до 1, сообщая нам вероятность того, что входные данные являются членами этой группы. Существует большое разнообразие в том, как эти модели обучаются и их свойства, но на высоком уровне они корректируют параметры, чтобы преобразовать ваши входные данные в группы подписчиков.

| **Дома/В гостях** | **# Фан.** |
|---|---|
| **Дома** | **100k** |
| **В гостях** | **50k** |
| **Дома** | **100k** |
| **Дома** | **99k** |
| **В гостях** | **50k** |
| **В гостях** | **10k** |

Рисунок 1.10 – Таблица соотношения кол-ва фанатов на двух типах матчей.

Это позволяет использовать методы, которые обычно так или иначе подсчитывают, увеличивая таким образом количество параметров на основе количества элементов, учитываемых в данных. Например, в контролируемой настройке непараметрическая модель может подсчитывать количество раз, когда определенный цвет уличного освещения заставляет автомобили «ехать». Рассмотрев лишь несколько примеров, эта модель сможет предсказать, что средние огни всегда (на 100%) вызывают «движение» автомобилей, а правильные огни только иногда (на 50%) вызывают «движение» автомобилей (рисунок 1.11).

Обратите внимание, что эта модель будет иметь 3 параметра, 3 отсчета, указывающих количество раз, когда каждый цветной свет включается, и автомобили «едут» (возможно, деленное на количество общих наблюдений). Если бы было 5 источников света, было бы 5 отсчетов (5 параметров). Что делает эту простую модель непараметрической, так это та черта, в которой количество параметров изменяется на основе данных (в данном случае, количества источников света). Это противоречит параметрическим моделям,

которые начинаются с заданного количества параметров и, что более важно, могут иметь более или менее параметры исключительно по усмотрению ученого, обучающего модель (независимо от данных).

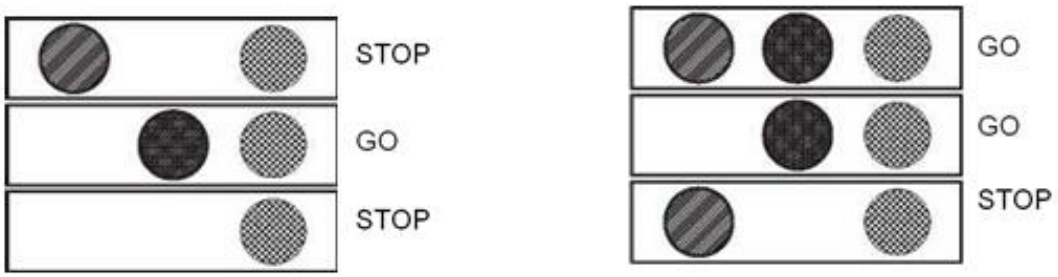


Рисунок 1.11 – Модель контролируемой непараметрической системы светофора

Пристальный взгляд может поставить под сомнение эту идею. В нашей параметрической модели, по-видимому, была ручка для каждого ввода данных. На самом деле, большинство параметрических моделей по-прежнему должны иметь какой-то ввод, основанный на количестве классов в данных. Таким образом, вы можете видеть, что между параметрическим и непараметрическим алгоритмами есть некоторая степень серости. Даже параметрические алгоритмы все еще в некоторой степени зависят от количества классов в данных, даже если они не учитывают явно шаблоны.

Возможно, это также освещает тот факт, что параметры на самом деле являются очень общим термином, относящимся только к набору чисел, используемых для моделирования шаблона (без каких-либо ограничений на использование этих чисел). Счетчики являются параметрами. Веса являются параметрами. Нормализованные варианты подсчета или веса являются параметрами. Коэффициенты корреляции могут быть параметрами. Это просто относится к набору чисел, используемых для моделирования шаблона. Как оказалось, Глубокое Обучение — это класс параметрических моделей. В этой книге мы не будем обсуждать непараметрические модели, но они представляют собой очень интересный и мощный класс алгоритмов.

### 1.2 Изучение интеллектуальных систем глубокого обучения

Люди и компьютеры по своей природе подходят для различных типов задач. Например, вычисление корня куба большого числа очень просто для компьютера, но чрезвычайно сложно для человека. С другой стороны, такая задача, как распознавание объектов на изображении

Это простой вопрос для человека, но традиционно он был очень сложным для алгоритма автоматического обучения. Только в последние годы глубокое обучение показало точность некоторых из этих задач, которая превышает задачу человека. На самом деле, последние результаты глубокого обучения

Алгоритмы, которые превосходят человеческие возможности в распознавании изображений (некоторые узкие задачи), не считались бы наиболее вероятными для большинства экспертов по компьютерному зрению еще 10 лет назад.

Многие архитектуры с глубоким обучением, которые показали такую необычную производительность, не создаются неразборчиво соединяющими вычислительными блоками. Превосходная производительность глубоких нейронных сетей отражает тот факт, что биологические нейронные сети также получают большую часть своей мощности от глубины. Кроме того, биологические сети связаны способами, которые мы не до конца понимаем. В тех немногих случаях, когда биологическая структура понимается на каком-то уровне, значительные прорывы были достигнуты путем создания искусственных нейронных сетей по этим направлениям. Классическим примером архитектуры такого типа является использование сверточной нейронной сети для распознавания изображений. Эта архитектура была вдохновлена экспериментами Хьюбела и Визеля в 1959 году по организации нейронов в зрительной коре кошки. Предшественником сверточной нейронной сети был неокогнитрон, который был непосредственно основан на этих результатах.

Структура нейрональных связей человека развивалась в течение миллионов лет для оптимизации управляемых выживанием показателей; выживание тесно связано с нашей способностью объединять ощущения и интуицию так, как это в настоящее время невозможно с машинами. Биологическая неврология — это область, которая все еще находится в зачаточном состоянии, и о том, как на самом деле работает мозг, известно лишь ограниченное количество. Поэтому справедливо предположить, что биологически вдохновленный успех сверточных нейронных сетей может быть воспроизведен в других условиях, поскольку мы узнаем больше о том, как работает человеческий мозг. Ключевое преимущество нейронных сетей по сравнению с традиционным машинным обучением состоит в том, что первый обеспечивает высокоуровневую абстракцию выражения семантического понимания областей данных с помощью выбора архитектурного дизайна в вычислительном графе. Второе преимущество заключается в том, что нейронные сети предоставляют простой способ настройки сложности модели путем добавления или удаления нейронов из архитектуры в соответствии с доступностью обучающих данных или вычислительной мощностью. Большая часть недавнего успеха нейронных сетей объясняется тем фактом, что возросшая доступность данных и вычислительная мощность современных компьютеров вышли за пределы традиционных алгоритмов машинного обучения, которые не в полной мере используют то, что сейчас возможно. Производительность традиционного машинного обучения в разы остается лучше для небольших наборов данных из-за большего выбора, большей простоты интерпретации моделей и тенденции к ручной интерпретации функций, которые включают в себя специфические для предметной области

знания. При ограниченных данных лучшие из очень широкого разнообразия моделей в машинном обучении обычно работают лучше, чем один класс моделей (например, нейронные сети). Это одна из причин, почему потенциал нейронных сетей не был реализован в первые годы.

Эра «больших данных» стала возможной благодаря достижениям в технологии сбора данных; практически все, что мы делаем сегодня, включая покупку товара, использование телефона или нажатие на сайт, где-то собирается и хранится. Кроме того, разработка мощных графических процессоров (GPU) позволила повысить эффективность обработки таких больших наборов данных. Эти достижения в значительной степени объясняют недавний успех глубокого обучения с использованием алгоритмов, которые лишь незначительно отличаются от версий, которые были доступны два десятилетия назад. Кроме того, эти недавние корректировки алгоритмов стали возможными благодаря повышенной скорости вычислений, поскольку сокращение времени выполнения позволяет проводить эффективное тестирование (и последующую алгоритмическую настройку). Если для тестирования алгоритма требуется месяц, можно протестировать не более двенадцати вариантов в год на одной аппаратной платформе. Эта ситуация исторически сдерживала интенсивные эксперименты, необходимые для настройки алгоритмов обучения нейронной сети. Быстрый прогресс, связанный с тремя столпами улучшения данных, вычислений и экспериментов, привел к все более оптимистичному взгляду на будущее глубокого обучения. Ожидается, что к концу этого столетия компьютеры будут способны обучать нейронные сети с таким же количеством нейронов, что и человеческий мозг. Хотя к настоящему времени трудно предсказать, какими будут истинные возможности искусственного интеллекта, наш опыт работы с компьютерным зрением должен подготовить нас к ожиданию неожиданного.

Виды моделей глубокого обучения можно разделить на несколько видов:

1. Модели внимания: люди не используют активно всю доступную им информацию из окружающей среды в любой момент времени. Скорее, они фокусируются на определенных частях данных, которые имеют отношение к поставленной задаче. Это биологическое понятие упоминается как внимание. Аналогичный принцип также может быть применен к приложениям искусственного интеллекта.

Модели с вниманием используют обучение с подкреплением (или другие методы), чтобы сосредоточиться на меньших порциях данных, которые имеют отношение к поставленной задаче. Такие методы недавно были использованы для повышения производительности.

2. Модели с избирательным доступом к внутренней памяти. Эти модели тесно связаны с моделями внимания, хотя разница заключается в том, что внимание сосредоточено в основном на определенных частях хранимых данных. Полезная аналогия - подумать о том, как люди обращаются к памяти

для выполнения конкретных задач. У людей есть огромное хранилище данных в ячейках памяти их мозга. Тем не менее, в любой данный момент, только небольшая его часть доступна, что соответствует поставленной задаче. Точно так же современные компьютеры имеют значительные объемы памяти, но компьютерные программы предназначены для выборочного и контролируемого доступа к ней с использованием переменных, которые являются механизмами косвенной адресации. Все нейронные сети имеют память в виде скрытых состояний. Однако он настолько тесно интегрирован с вычислениями, что трудно отделить доступ к данным от вычислений. Управляя чтением и записью во внутреннюю память нейронной сети более избирательно и явно вводя понятие механизмов адресации, получающаяся сеть выполняет вычисления, которые более точно отражают человеческий стиль программирования. Часто такие сети имеют лучшую мощность обобщения, чем более традиционные нейронные сети при выполнении прогнозов для данных вне выборки. Можно также рассматривать выборочный доступ к памяти как применение внутренней формы внимания к памяти нейронной сети. Результирующая архитектура называется сетью памяти или нейронной машиной Тьюринга.

3. Генеративные состязательные сети. Генеративные состязательные сети предназначены для создания генеративных моделей данных из выборок. Эти сети могут создавать реалистично выглядящие образцы из данных, используя две противоборствующие сети. Одна сеть генерирует синтетические сэмплы (генератор), а другая (которая является дискриминатором) классифицирует смесь оригинальных экземпляров и сгенерированных сэмплов как реальные или синтетические. Состязательная игра приводит к улучшению генератора с течением времени, пока дискриминатор не сможет больше различать реальные и поддельные сэмплы. Кроме того, путем согласования с конкретным типом контекста (например, подписью к изображению) также можно направлять создание конкретных типов желаемых выборок.

Механизмам внимания часто приходится принимать трудные решения относительно конкретных частей данных, к которым нужно обращаться. Можно рассматривать этот выбор аналогично вариантам, с которыми сталкивается алгоритм обучения с подкреплением. Некоторые из методов, используемых для построения моделей, основанных на внимании, в значительной степени основаны на обучении с подкреплением, а другие - нет. Нейронные машины Тьюринга относятся к тесно связанному классу архитектур, называемых сетями памяти. В последнее время они показали многообещающие результаты при создании систем ответов на вопросы, хотя результаты все еще довольно примитивны. Построение нейронной машины Тьюринга можно считать воротами ко многим возможностям искусственного интеллекта, которые еще не полностью реализованы. Как обычно в историческом опыте с нейронными сетями, больше данных и вычислительных

мощностей будут играть важную роль в реализации этих обещаний в реальность.

Совершенно другим способом обучения является способ конкурентного обучения, при котором нейроны соревнуются за право реагировать на подмножество входных данных. Веса изменяются в зависимости от победителя этого конкурса. Этот подход представляет собой вариант изучения, и полезен для неконтролируемых учебных приложений, таких как кластеризация, уменьшение размерности и сжатие.

Рассматривая системы глубокого обучения, нужно уметь сравнивать результат прогноза и учиться на полученном опыте.

Сравнение. Измерение того, насколько наш прогноз «пропустил». После того, как мы сделали прогноз, следующим шагом для изучения является оценка того, насколько хорошо мы справились. Возможно, это может показаться довольно простой концепцией, но мы в конечном итоге обнаружим, что придумать хороший способ измерить ошибку является одним из наиболее важных и сложных предметов глубокого обучения.

На самом деле, существует много свойств «ошибки измерения», которые вы, вероятно, делали всю свою жизнь, не осознавая этого. Возможно, вы (или ваш знакомый) усиливаете большие ошибки, игнорируя при этом очень маленькие. В этой главе мы узнаем, как математически научить нашу сеть делать это. Кроме того (и это может показаться слишком простым что важно), мы узнаем, что ошибка всегда положительна! Мы рассмотрим аналогию «лучника», поражающего цель. Является ли он слишком низким и дюймовым или слишком высоким на дюйм, ошибка все равно составляет всего 1 дюйм! В нашем шаге «Сравнение» нейронной сети мы хотим учитывать эти виды свойств при измерении погрешности.

Мы рассмотрим только один, очень простой способ измерения ошибки, называемый «средняя квадратическая ошибка». Тем не менее, это всего лишь один из многих способов оценки точности вашей нейронной сети.

Как заключительная мысль, этот шаг даст нам представление о том, «сколько мы пропустили», но этого недостаточно, чтобы иметь возможность учиться. Выход нашей логики сравнения будет просто сигналом типа «горячий или холодный». Учитывая некоторый прогноз, мы вычислим меру ошибки, которая скажет «много» или «немного». Это не скажет нам, почему мы пропустили, в каком направлении мы пропустили, или что мы должны сделать, чтобы это исправить. Это более или менее просто говорит «большая мисс», «маленькая мисс» или «идеальный прогноз».

То, что мы делаем с нашей ошибкой, фиксируется на следующем шаге.

«Обучение» принимает нашу ошибку и сообщает каждому весу, как он может измениться, чтобы уменьшить его.

Обучение — это «атрибуция ошибок» или искусство выяснения того, как каждый вес сыграл свою роль в создании ошибки.

В конце концов, это приведет к вычислению числа для каждого из наших весов. Это число будет представлять, как этот вес должен быть выше

или ниже, чтобы уменьшить ошибку. Затем мы переместим вес в соответствии с этим числом, и все будет готово. Мы можем изучить пример, при котором существует возможность измерения ошибки (рисунок 1.12).

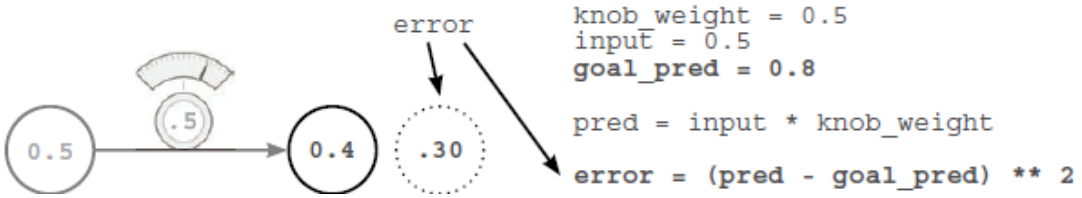


Рисунок 1.12 – Расчет среднеквадратичной ошибки

Значение переменной goal_pred. Очень похоже на ввод, это число, которое мы записали где-то в реальном мире, но обычно это трудно заметить, например, «процент людей, которые носили спортивные костюмы» с учетом температуры или «действительно ли DID выиграли домашнюю игру» учитывая его средний уровень.

Здесь следует обозначить такие вопросы как: почему ошибка возводится в квадрат; данные о лучнике, поражающем цель; когда он ошибается на 2 дюйма, сколько он пропустил; когда он на два дюйма ниже, сколько он пропустил. Оба раза он пропустил только 2 дюйма. Основная причина, по которой мы утверждаем, «сколько мы пропустили», заключается в том, что это заставляет результат быть положительным. pred-goal_pred может быть отрицательным в некоторых ситуациях ... в отличие от реальной ошибки.

Квадрат делает большие ошибки (> 1) больше, а маленькие (<1) меньше. Это довольно странный способ измерения ошибок, но оказывается, что увеличение больших ошибок и уменьшение мелких ошибок на самом деле нормально. Позже мы будем использовать эту ошибку, чтобы помочь сети учиться, и мы бы предпочли, чтобы она обращала внимание на большие ошибки и не особо волновалась о мелких.

Ошибка измерения упрощает проблему.

Цель обучения нашей нейронной сети - делать правильные прогнозы. Это то, что мы хотим. И в самом прагматичном мире (как упомянуто в предыдущей главе) мы хотим, чтобы сеть принимала входные данные, которые мы можем легко вычислить (сегодняшняя цена акций), и предсказывать вещи, которые трудно вычислить (завтрашняя цена акций). Вот что делает нейронную сеть полезной. Оказывается, что «изменение knob_weight для того, чтобы сеть правильно предсказывала goal_prediction», немного сложнее, чем «изменение knob_weight для создания ошибки == 0». Есть что-то более лаконичное в рассмотрении проблемы таким образом. В конечном итоге оба эти утверждения говорят об одном и том же, но попытка получить ошибку до 0 кажется немного более простой.

## 1.3 Принцип работы и основы нейронных сетей

Искусственные нейронные сети являются популярными методами машинного обучения, которые имитируют механизм обучения у биологических организмов. Нервная система человека содержит клетки, которые называются нейронами. Нейроны связаны друг с другом с помощью аксонов и дендритов, а соединительные области между аксонами и дендритами называются синапсами. Эти соединения показаны на рисунке 13. Сила синаптических связей часто меняется в ответ на внешние раздражители. Это изменение, как обучение происходит в живых организмах.

Этот биологический механизм моделируется в искусственных нейронных сетях, которые содержат вычислительные единицы, которые называются нейронами. В этой книге мы будем использовать термин «нейронные сети» для обозначения искусственных нейронных сетей, а не биологических. Вычислительные единицы связаны друг с другом с помощью весов, которые служат одинаково.

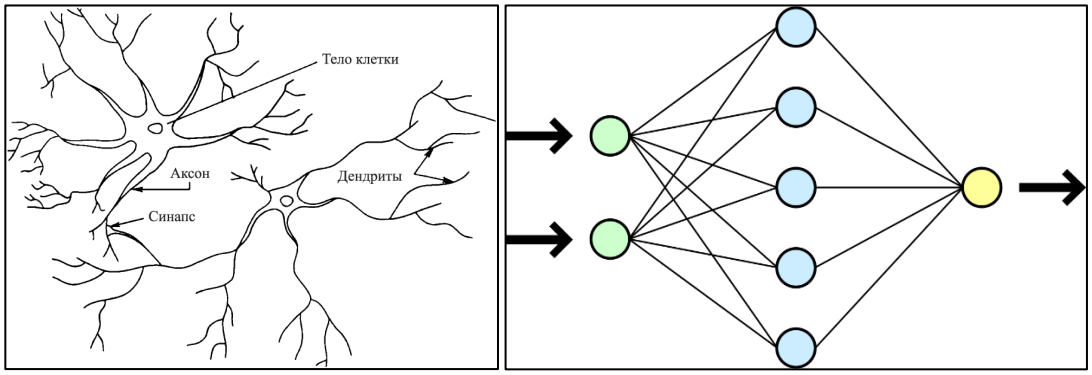


Рисунок 1.13 – а) биологическая нейронная сеть; б) искусственная нейронная сеть

Этот биологический механизм моделируется в искусственных нейронных сетях, которые содержат вычислительные единицы, которые называются нейронами. В этой книге мы будем использовать термин «нейронные сети» для обозначения искусственных нейронных сетей, а не биологических. Вычислительные единицы связаны друг с другом с помощью весов, которые выполняют ту же роль, что и силы синаптических связей в биологических организмах. Каждый вход в нейрон масштабируется с весом, который влияет на функцию, вычисленную в этой единице. Эта архитектура показана на рисунке 1.13 (б). Искусственная нейронная сеть вычисляет функцию входных данных, распространяя вычисленные значения из входных нейронов в выходные нейроны и используя веса в качестве промежуточных параметров. Обучение происходит путем изменения весов, соединяющих нейроны. Так же, как внешние стимулы необходимы для обучения в биологических организмах, внешний стимул в искусственных нейронных сетях обеспечивается обучающими данными, содержащими примеры пар ввода-вывода функции, которую необходимо изучить. Например, обучающие

данные могут содержать пиксельные представления изображений (входные данные) и их аннотированные метки (например, морковь, банан) в качестве выходных данных. Эти пары обучающих данных подаются в нейронную сеть с использованием входных представлений для прогнозирования выходных меток. Обучающие данные обеспечивают обратную связь с правильностью весов в нейронной сети в зависимости от того, насколько хорошо прогнозируемый выходной сигнал (например, вероятность появления моркови) для конкретного входа соответствует аннотированной выходной метке в обучающих данных. Можно рассматривать ошибки, допущенные нейронной сетью при вычислении функции, как своего рода неприятную обратную связь в биологическом организме, приводящую к корректировке синаптических сил.

Точно так же веса между нейронами настраиваются в нейронной сети в ответ на ошибки прогнозирования. Целью изменения весов является изменение вычисляемой функции, чтобы сделать прогнозы более правильными в будущих итерациях. Поэтому веса тщательно изменяются математически обоснованным способом, чтобы уменьшить ошибку в вычислениях в этом примере. Путем последовательной корректировки весов между нейронами по множеству пар ввода-вывода функция, вычисляемая нейронной сетью, уточняется с течением времени, чтобы обеспечить более точные прогнозы. Следовательно, если нейронная сеть обучена множеству различных изображений бананов, она в конечном итоге сможет правильно распознать банан по изображению, которого он не видел раньше. Эта способность точных вычислений функций невидимых входов путем обучения конечному набору пар ввода-вывода называется обобщением модели. Основная полезность всех моделей машинного обучения заключается в их способности обобщать свое обучение от видимых обучающих данных до невиданных примеров.

Биологическое сравнение часто критикуют как очень плохую карикатуру на работу человеческого мозга; тем не менее, принципы нейронауки часто были полезны при разработке архитектуры нейронной сети. Другое мнение состоит в том, что нейронные сети строятся как абстракции более высокого уровня классических моделей, которые обычно используются в машинном обучении. Фактически, самые основные единицы вычислений в нейронной сети основаны на традиционных алгоритмах машинного обучения, таких как регрессия методом наименьших квадратов и логистическая регрессия. Нейронные сети получают свои возможности, собирая множество таких базовых единиц и совместно изучая веса различных единиц, чтобы минимизировать ошибку прогнозирования. С этой точки зрения нейронная сеть может рассматриваться как вычислительный граф элементарных единиц, в которых большая мощность приобретается путем их соединения особым образом. Когда нейронная сеть используется в ее самой базовой форме, без объединения нескольких единиц, алгоритмы обучения часто сводятся к классическим моделям машинного обучения. Реальная сила нейронной

модели над классическими методами раскрывается, когда эти элементарные вычислительные единицы объединяются, и веса элементарных моделей обучаются с использованием их зависимостей друг от друга. Комбинируя несколько единиц, можно повысить способность модели изучать более сложные функции данных, чем присущи элементарные модели базового машинного обучения. Способ, которым эти единицы объединяются, также играет роль во власти архитектуры и требует некоторого понимания и понимания от аналитика. Кроме того, для обучения большего числа весов в этих расширенных вычислительных графиках также требуются достаточные обучающие данные (рисунок 1.14).

Базовая архитектура нейронных сетей. В этом разделе мы рассмотрим однослойные и многослойные нейронные сети. В однослойной сети набор входов напрямую отображается на выход с использованием обобщенного варианта линейной функции. Это простое создание нейронной сети также называется персептроном. В многослойных нейронных сетях нейроны расположены слоистым образом, в котором входной и выходной слои разделены группой скрытых слоев.

Эта послойная архитектура нейронной сети также упоминается как сеть прямой связи. В этом разделе будут обсуждаться как однослойные, так и многослойные сети на основе математической модели (персептрона) (рисунок 1.14).


Рисунок 1.14 – Сравнение точности типичного алгоритма машинного обучения и нейросети.

Простейшая нейронная сеть называется персептроном. Эта нейронная сеть содержит один входной слой и выходной узел. Базовая архитектура персептрона показана на рисунке 1.14. Рассмотрим ситуацию, когда каждый обучающий экземпляр имеет форму (X, y), где каждый X = [x1,. , , xd] содержит d характерных переменных, а y ∈ {−1, + 1} содержит наблюдаемое значение двоичной переменной класса.

Под «наблюдаемой ценностью» мы понимаем тот факт, что она предоставляется нам как часть данных обучения, и наша цель состоит в том, чтобы предсказать переменную класса для случаев, в которых она не наблюдается.

Например, в приложении для обнаружения мошенничества с кредитными картами функции могут представлять различные свойства набора транзакций по кредитным картам (например, количество и частота транзакций), а переменная класса может представлять, является ли этот набор транзакций мошенническим или нет.

Понятно, что в этом типе приложения могут быть исторические случаи, в которых наблюдалась переменная класса, и другие (текущие) случаи, в которых переменная класса еще не наблюдалась, но должна быть предсказана.

Входной слой содержит d узлов, которые передают d функций $\bar{X}$ = [$x_1$… $x_d$] с ребрами веса$\bar{W}$= [$w_1$… $w_d$] к выходному узлу. Входной слой не выполняет никаких вычислений сам по себе. Линейная функция $\bar{W} * \bar{X} = \sum_{i=1}^{d} w_i x_i$ вычисляется на выходном узле. Впоследствии знак этого действительного значения используется для прогнозирования зависимой переменной $\bar{X}$.

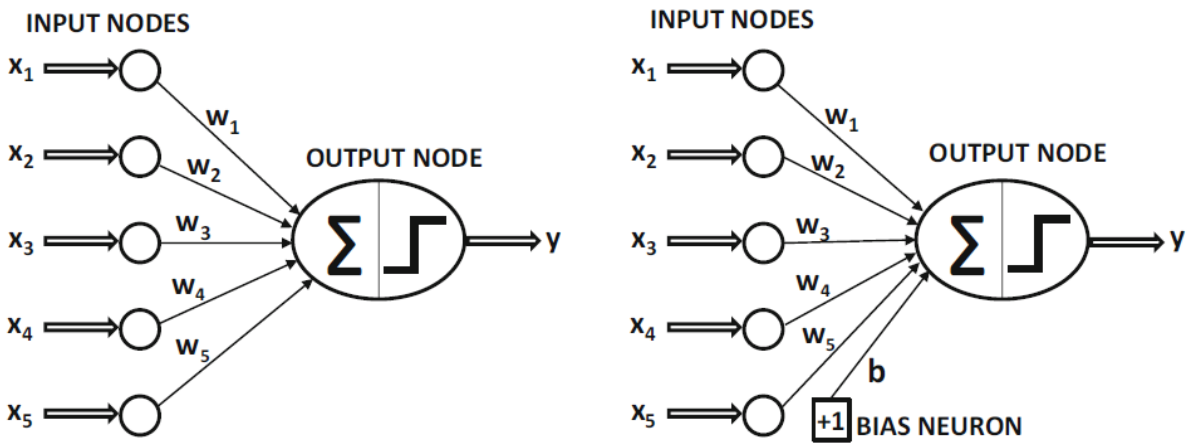


Рисунок 1.15 - Основная архитектура персептрона.
(а) персептрон без смещения; б) персептрон со смещением)

Поэтому прогноз y вычисляется следующим образом:

$$\hat{y} = sign\{\bar{W} * \bar{X}\} = sign\left\{\sum_{j=1}^{d} w_j x_j\right\} \quad (1.1)$$

Функция знака отображает действительное значение в +1 или -1, что подходит для двоичной классификации. Обратите внимание на окружность в верхней части переменной y, чтобы указать, что это прогнозируемое значение, а не фактическое значение. Следовательно, ошибка прогноза $E(\bar{X}) = y - \hat{y}$,

который является одним из значений, взятых из множества {−2, 0, +2}. В случаях, когда значение ошибки E (X) отлично от нуля, веса в нейронной сети необходимо обновлять в (отрицательном) направлении градиента ошибки. Как мы увидим позже, этот процесс аналогичен тому, который используется в различных типах линейных моделей в машинном обучении. Несмотря на сходство персептрона с традиционными моделями машинного обучения, его интерпретация как вычислительной единицы очень полезна, поскольку позволяет нам собирать несколько единиц для создания гораздо более мощных моделей, чем те, которые доступны в традиционном машинном обучении.

Архитектура персептрона показана на рисунке 1.15, в котором один входной слой передает функции на выходной узел. Ребра от входа до выхода содержат веса w1…wd, с которым функции умножаются и добавляются в выходной узел.

Впоследствии, функция знака применяется для преобразования агрегированного значения в метку класса. Функция знака выполняет роль функции активации. Различные варианты функций активации могут использоваться для моделирования различных типов моделей, используемых в машинном обучении, например регрессия наименьших квадратов с числовыми целями, машина опорных векторов или классификатор логистической регрессии. Большинство базовых моделей машинного обучения легко представить в виде простых архитектур нейронных сетей. Это полезное упражнение для моделирования традиционных методов машинного обучения в качестве нейронных архитектур, поскольку оно дает более четкую картину того, как глубокое обучение обобщает традиционное машинное обучение. Эта точка зрения подробно рассматривается в главе 2. Примечательно, что персептрон содержит два слоя, хотя входной слой не выполняет никаких вычислений и только передает значения признаков. Входной слой не учитывается при подсчете количества слоев в нейронной сети. Поскольку персептрон содержит один вычислительный уровень, он считается однослойной сетью.

Во многих настройках существует инвариантная часть прогноза, которая называется смещением. Например, рассмотрим параметр, в котором переменные объекта имеют среднее значение по центру, но среднее значение предсказания двоичного класса из {-1, +1} не равно 0. Это будет иметь место в ситуациях, когда распределение двоичного класса сильно несбалансированным. В таком случае вышеупомянутого подхода недостаточно для прогнозирования. Нам нужно включить дополнительную переменную смещения b, которая фиксирует эту инвариантную часть прогноза:

$$\hat{y}=\text{sign}\{\overline{W}*\overline{X}+b\}=\text{sign}\left\{\sum_{j=1}^{d} w_j x_j+b\right\} \qquad (1.2)$$

Смещение может быть учтено как вес ребра с помощью нейрона смещения. Это достигается путем добавления нейрона, который всегда передает значение 1 на выходной узел. Вес ребра, соединяющего нейрон смещения с выходным узлом, обеспечивает переменную смещения.

Пример нейрона смещения показан на рисунке 1.15 (б). Другой подход, который хорошо работает с однослойными архитектурами, состоит в том, чтобы использовать прием разработки функций, при котором дополнительная функция создается с постоянным значением 1. Коэффициент этой функции обеспечивает смещение, и тогда можно работать с уравнением (1.1) На протяжении всех расчетов смещения не будут использоваться (для простоты архитектурных представлений), потому что они могут быть включены с нейронами смещения. Детали обучающих алгоритмов остаются такими же, просто обрабатывая нейроны смещения, как и любой другой нейрон с фиксированным значением активации, равным 1. Следовательно, следующее будет работать с прогнозирующим допущением уравнения (1.2), которое явно не использует смещения.

В то время, когда алгоритм персептрона был предложен Розенблаттом, эти оптимизации были выполнены эвристическим способом с использованием реальных аппаратных схем, и он не был представлен с точки зрения формального понятия оптимизации в машинном обучении (как это принято сегодня) , Однако цель всегда заключалась в том, чтобы минимизировать ошибку в прогнозировании, даже если формальная оптимизационная формулировка не была представлена. Таким образом, алгоритм персептрона был эвристически спроектирован, чтобы минимизировать количество ошибочных классификаций, и были доступны доказательства сходимости, которые обеспечивали правильность алгоритма обучения в упрощенных настройках.

Следовательно, мы все еще можем записать эвристически мотивированную цель алгоритма персептрона в форме наименьших квадратов по отношению ко всем обучающим экземплярам в наборе данных D, содержащем пары меток признаков:

$$Minimize_{\bar{W}} L = \sum_{(\bar{X}y)\in D} (y - \hat{y})^2 = \sum_{(\bar{X}y)\in D} (y - sign\{\bar{W} * \bar{X}\})^2 \qquad (1.3)$$

Этот тип целевой функции минимизации также называется функцией потерь. Как мы увидим позже, почти все алгоритмы обучения нейронной сети формулируются с использованием функции потерь. Как мы узнали исходя из выше сказанного, эта функция потерь во многом похожа на регрессию наименьших квадратов. Однако последний определен для непрерывных целевых переменных, и соответствующие потери являются гладкой и непрерывной функцией переменных. С другой стороны, для формы

наименьших квадратов целевой функции знаковая функция недифференцируема с скачкообразными переменами в определенных точках. Кроме того, функция знака принимает постоянные значения на больших участках домена, и поэтому точный градиент принимает нулевые значения в дифференцируемых точках. Это приводит к потере поверхности, похожей на лестницу, которая не подходит для градиентного спуска. Алгоритм персептрона (неявно) использует плавное приближение градиента этой целевой функции по отношению к каждому примеру:

$$\triangle L_{smooth} = \sum_{(\bar{X}y)\in D} (y - \hat{y})\bar{X} \tag{1.4}$$

Обратите внимание, что приведенный выше градиент не является истинным градиентом ступенчатой поверхности (эвристической) целевой функции, которая не обеспечивает полезных градиентов. Поэтому лестница сглаживается в наклонную поверхность, определяемую критерием персептрона. Примечательно, что такие понятия, как «критерий персептрона», были предложены позже, чем оригинальная статья Розенблатта, для объяснения эвристических шагов градиентного спуска. Сейчас мы будем предполагать, что алгоритм персептрона оптимизирует некоторую неизвестную гладкую функцию с использованием градиентного спуска.

Хотя указанная выше целевая функция определена для всех обучающих данных, алгоритм обучения нейронных сетей работает путем подачи каждого экземпляра входных данных X в сеть один за другим (или небольшими партиями) для создания прогноза $\hat{y}$. Затем веса обновляются на основе значения ошибки $\mathrm{E}(\bar{\mathrm{X}}) = (y - \hat{y})$. В частности, когда точка X данных подается в сеть, весовой вектор $\bar{W}$ обновляется следующим образом:

$$\bar{W} \Leftarrow \bar{W} + \alpha E(y - \hat{y})\bar{X} \tag{1.5}$$

Параметр α регулирует скорость обучения нейронной сети. Алгоритм персептрона многократно циклически перебирает все обучающие примеры в случайном порядке и итеративно корректирует веса до достижения сходимости. Одна точка обучающих данных может циклически повторяться много раз. Каждый такой цикл называется эпохой. Также можно записать обновление градиента в терминах ошибки $\mathrm{E}(\bar{\mathrm{X}}) = (y - \hat{y})$ следующим образом:

$$\bar{W} \Leftarrow \bar{W} + \alpha E(\bar{X})\bar{X} \tag{1.6}$$

Базовый алгоритм персептрона можно считать стохастическим методом градиентного спуска, который неявно минимизирует квадратичную ошибку прогнозирования, выполняя обновления градиентного спуска относительно

случайно выбранных тренировочных точек. Предполагается, что нейронная сеть циклически перебирает точки в случайном порядке во время обучения и меняет веса с целью уменьшения ошибки прогнозирования в этой точке. Из уравнения (1.5) легко увидеть, что ненулевые обновления производятся для весов только тогда, когда $y = \hat{y}$, что происходит только тогда, когда при прогнозировании допускаются ошибки. При мини-пакетном стохастическом спуске градиента вышеупомянутые обновления уравнения (1.5) реализуются в случайно выбранном подмножестве обучающих точек S (рисунок 1.16):

$$\bar{W} \Leftarrow \bar{W} + \alpha \sum_{(\bar{X}y)\in D} E(\bar{X})\bar{X} \tag{1.7}$$

Интересная особенность персептрона состоит в том, что можно установить скорость обучения α равной 1, поскольку скорость обучения только масштабирует веса.

Тип модели, предложенной в персептроне, представляет собой линейную модель, в которой уравнение $\bar{W} * \bar{X} = 0$ определяет линейную гиперплоскость. Здесь W = (w1 ... wd) — это d-мерный вектор, нормальный к гиперплоскости. Кроме того, значение $\bar{W}\bar{X}$ является положительным для значений $\bar{X}$ на одной стороне гиперплоскости, и оно является отрицательным для значений $\bar{X}$ на другой стороне. Этот тип модели работает особенно хорошо, когда данные линейно разделимы. Примеры линейно разделимых и неразделимых данных показаны на рисунке 1.16.

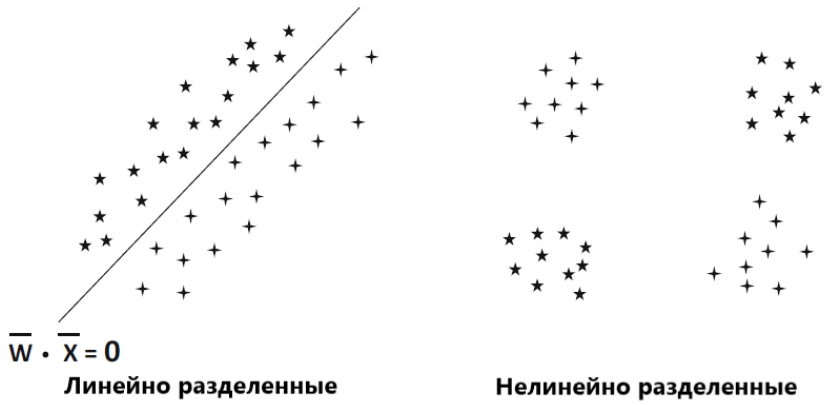


Рисунок 1.16 - Примеры линейно разделимых и неразделимых данных в двух классах

Алгоритм персептрона хорошо классифицирует наборы данных, подобные показанному в левой части рисунка 1.16, когда данные линейно разделимы. С другой стороны, он плохо работает с наборами данных, подобными тем, которые показаны в правой части рисунка 1.16. В этом примере показано ограничение моделирования, присущее персептрону, что требует использования более сложных нейронных архитектур.

Поскольку оригинальный алгоритм персептрона был предложен как эвристическая минимизация ошибок классификации, было особенно важно показать, что алгоритм сходится к разумным решениям в некоторых особых случаях. В этом контексте было показано, что алгоритм персептрона всегда сходится, чтобы обеспечить нулевую ошибку в обучающих данных, когда данные линейно разделимы. Однако алгоритм персептрона не гарантированно сходится в случаях, когда данные не являются линейно разделимыми. По причинам, обсуждаемым в следующем разделе, персептрон может иногда прийти к очень плохому решению с данными, которые не могут быть линейно разделены (по сравнению со многими другими алгоритмами обучения).

Как обсуждалось ранее, в первоначальной работе Перцептрона Розенблатта формально не предлагались функции потерь. В те годы эти реализации были достигнуты с использованием реальных аппаратных схем. Первоначальный персептрон Mark I должен был быть машиной, а не алгоритмом, и для его создания использовалось специальное оборудование (рисунок 1.17).

Общая цель состояла в том, чтобы минимизировать количество ошибок классификации с помощью эвристического процесса обновления (в аппаратном обеспечении), который изменял веса в «правильном» направлении всякий раз, когда были допущены ошибки. Это эвристическое обновление сильно напоминало градиентный спуск, но оно не было получено как метод градиентного спуска. Градиентный спуск определяется только для гладких функций потерь в алгоритмических настройках, тогда как аппаратно-ориентированный подход был разработан в более эвристический способ с двоичными выходами. Многие из бинарных и схемотехнических принципов были унаследованы от модели нейрона МакКаллох-Питтс. К сожалению, двоичные сигналы не подвержены постоянной оптимизации.

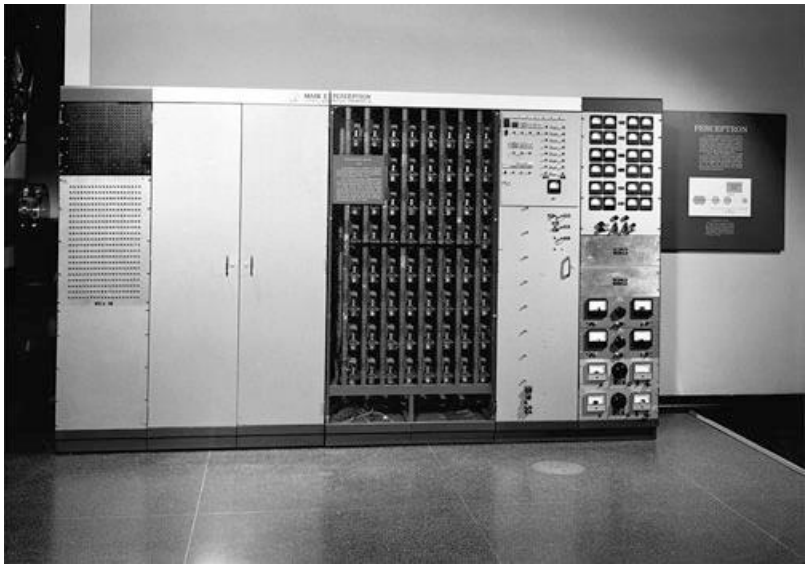

Рисунок 1.17 - Перцептрон марки I, построенный в 1958 году.

Теперь возможно найти гладкую функцию потерь, градиент которой оказывается обновлением персептрона. Число ошибок классификации в задаче двоичной классификации может быть записано в виде функции потерь 0/1 для обучающей точки данных $(\bar{X}_\iota, y_i)$ следующим образом:

$$L_i^{0/1} = \frac{1}{2}(y_i - sign\{\bar{W} * \bar{X}\})^2 = 1 - y_i * sign\{\bar{W} * \bar{X}\} \qquad (1.8)$$

Упрощение с правой стороны вышеуказанной целевой функции получается установкой $y_i^2$ и знака $sign\{\bar{W} * \bar{X}\}^2$ на 1, поскольку они получаются путем возведения в квадрат значения, полученного из $\{-1, +1\}$. Однако эта целевая функция не дифференцируема, поскольку имеет лестничную форму, особенно когда она добавляется в несколько точек. Обратите внимание, что в потере 0/1 выше преобладает член $-y_i\{\bar{W}*\bar{X}\}$, в котором знаковая функция вызывает большинство проблем, связанных с недифференцируемостью. Поскольку нейронные сети определяются с помощью градиентной оптимизации, нам необходимо определить гладкую целевую функцию, которая отвечает за обновления перцептрона. Можно показать, что обновления персептрона неявно оптимизируют критерий персептрона. Эта целевая функция определяется путем сброса функции знака в приведенной выше потере 0/1 и установки отрицательных значений в 0, чтобы обрабатывать все правильные прогнозы единообразно и без потерь:

$$L_i = max\{-y_i(\bar{W} * \bar{X}).0\} \qquad (1.9)$$

$\bar{W} \Leftarrow \overline{W} - \alpha \nabla W L_i$, Модифицированная функция потерь, позволяющая вычислять градиент недифференцируемой функции, также называется сглаженной суррогатной функцией потерь. Почти во всех методах обучения, основанных на непрерывной оптимизации (таких как нейронные сети) с дискретными выходными данными (например, метками классов), используется некоторый тип сглаженной функции суррогатных потерь (рисунок 1.18).

Рисунок 1.18 - Критерий персептрона против потери угла

Хотя вышеупомянутый критерий персептрона был обратным образом спроектирован путем обратной работы с обновлениями персептрона, природа этой функции потерь раскрывает некоторые слабые стороны обновлений в исходном алгоритме. Интересное наблюдение относительно критерия персептрона состоит в том, что можно установить $\bar{W}$ в нулевой вектор независимо от набора обучающих данных, чтобы получить оптимальное значение потерь 0. Несмотря на этот факт, обновления персептрона продолжают сходиться к четкому разделителю между двумя классами в линейно разделимых случаях; в конце концов, разделитель между двумя классами также обеспечивает значение потерь 0. Однако поведение данных, которые не являются линейно разделимыми, довольно произвольно, и полученное решение иногда даже не является хорошим приближенным разделителем классов. Прямая чувствительность потери к величине вектора веса может ослабить цель разделения классов; обновления могут значительно ухудшить количество ошибочных классификаций, одновременно увеличивая потери. Это пример того, как функции суррогатных потерь могут иногда не полностью достигать поставленных целей. Из-за этого факта этот подход не стабилен и может дать решения широкого спектра качества. Поэтому было предложено несколько вариантов алгоритма обучения для неразделимых данных, и естественным подходом является всегда отслеживать лучшее решение с точки зрения количества ошибочных классификаций. Такой подход всегда держать лучшее решение в своем «кармане» называется карманным алгоритмом. Другой высокоэффективный вариант включает в себя понятие запаса в функции потерь, что создает алгоритм, идентичный линейному механизму опорных векторов. По этой причине, линейные опорные векторы также упоминаются как персептроне оптимальной стабильности.

## 1.4 Глубокое обучение в технологиях компьютерного зрения

Проблема поиска шаблонов в данных является фундаментальной и имеет долгую и успешную историю. Например, обширные астрономические наблюдения Тихо Браге в 16 веке позволили Йоханнесу Кеплеру открыть эмпирические законы движения планет, что, в свою очередь, послужило трамплином для развития классической механики. Точно так же открытие закономерностей в атомных спектрах сыграло ключевую роль в развитии и проверке квантовой физики в начале двадцатого века. Область распознавания образов связана с автоматическим обнаружением закономерностей в данных с помощью компьютерных алгоритмов и с помощью этих закономерностей для выполнения таких действий, как классификация данных по различным категориям. Рассмотрим пример распознавания рукописных цифр, показанный на рисунке 18. Каждая цифра соответствует изображению 28 × 28 пикселей и может быть представлена вектором x, содержащим 784 действительных числа. Цель состоит в том, чтобы построить машину, которая

будет принимать такой вектор x в качестве входных данных и который будет производить идентичность цифры 0, …, 9 в качестве выхода.

Это нетривиальная проблема из-за широкой изменчивости почерка. Эту проблему можно решить, используя правила ручной работы или эвристику для различения цифр на основе форм штрихов, но на практике такой подход приводит к быстрому распространению правил и исключений из правил и т.д. и неизменно дает плохие результаты (рисунок 1.19).

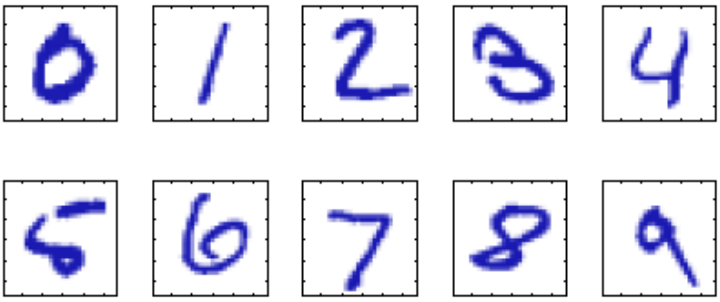

Рисунок 1.19 – Пример написанных от руки цифр

Намного лучшие результаты можно получить, приняв подход машинного обучения, в котором большой набор из N цифр $\{x_1, \ldots, x_N\}$, называемый обучающим набором, используется для настройки параметров адаптивной модели. Категории цифр в обучающем наборе известны заранее, обычно путем их индивидуального осмотра и ручной маркировки. Мы можем выразить категорию цифры, используя целевой вектор t, который представляет собой идентичность соответствующей цифры. Подходящие методы для представления категорий в виде векторов будут обсуждаться позже. Обратите внимание, что существует один такой целевой вектор t для каждого цифрового изображения x.

Результат запуска алгоритма машинного обучения может быть выражен как функция y (x), которая принимает новое цифровое изображение x в качестве входных данных и генерирует выходной вектор y, закодированный так же, как целевые векторы. Точная форма функции y (x) определяется на этапе обучения, также известном как этап обучения, на основе данных обучения. Как только модель обучена, она может определить идентичность новых цифровых изображений, которые, как говорят, составляют тестовый набор. Способность правильно классифицировать новые примеры, которые отличаются от тех, которые используются для обучения, называется обобщением. В практических применениях изменчивость входных векторов будет такой, что обучающие данные могут составлять лишь крошечную долю всех возможных входных векторов, и поэтому обобщение является центральной целью распознавания образов.

Для большинства практических применений исходные входные переменные обычно предварительно обрабатываются, чтобы преобразовать их в какое-то новое пространство переменных, где, как ожидается, проблему

распознавания образов будет легче решить. Например, в задаче распознавания цифр изображения цифр обычно переводятся и масштабируются так, чтобы каждая цифра содержалась в рамке фиксированного размера. Это значительно уменьшает изменчивость в каждом классе цифр, потому что расположение и масштаб всех цифр теперь одинаковы, что значительно облегчает последующему алгоритму распознавания образов различать разные классы. Эта стадия предварительной обработки иногда также называется извлечением признаков. Обратите внимание, что новые тестовые данные должны быть предварительно обработаны с использованием тех же шагов, что и данные обучения.

Предварительная обработка также может выполняться для ускорения вычислений. Например, если целью является обнаружение лица в реальном времени в видеопотоке с высоким разрешением, компьютер должен обрабатывать огромное количество пикселей в секунду, и представление их непосредственно алгоритму распознавания сложных образов может быть вычислительно неосуществимым. Вместо этого цель состоит в том, чтобы найти полезные функции, которые можно быстро вычислить, но которые также сохраняют полезную дискриминационную информацию, позволяющую отличить лица от лиц. Эти функции затем используются в качестве входных данных для алгоритма распознавания образов. Например, среднее значение интенсивности изображения по прямоугольному субрегиону может быть оценено чрезвычайно эффективно, и набор таких функций может оказаться очень эффективным при быстром обнаружении лица. Поскольку количество таких признаков меньше количества пикселей, этот вид предварительной обработки представляет собой форму уменьшения размерности. Во время предварительной обработки необходимо соблюдать осторожность, поскольку часто информация отбрасывается, и если эта информация важна для решения проблемы, то может пострадать общая точность системы.

Приложения, в которых обучающие данные содержат примеры входных векторов вместе с соответствующими им целевыми векторами, известны как контролируемые проблемы обучения.

Случаи, такие как пример распознавания цифр, в котором цель состоит в том, чтобы назначить каждый входной вектор одной из конечного числа дискретных категорий, называются проблемами классификации. Если желаемый результат состоит из одной или нескольких непрерывных переменных, то задача называется регрессией. Примером проблемы регрессии может быть прогнозирование выхода в процессе химического производства, в котором входные данные состоят из концентраций реагентов, температуры и давления.

В других задачах распознавания образов обучающие данные состоят из набора входных векторов x без каких-либо соответствующих целевых значений. Целью таких неконтролируемых проблем обучения может быть обнаружение групп схожих примеров в данных, распределение данных во входном пространстве, известное как оценка плотности, или проецирование

данных из многомерного пространства в два или три измерения. с целью визуализации.

Наконец, методика обучения с подкреплением связана с проблемой поиска подходящих действий в данной ситуации, чтобы максимизировать вознаграждение. Здесь алгоритм обучения не дает примеров оптимальных результатов, в отличие от контролируемого обучения, но вместо этого должен обнаруживать их методом проб и ошибок. Обычно существует последовательность состояний и действий, в которых алгоритм обучения взаимодействует со своей средой. Во многих случаях текущее действие не только влияет на немедленное вознаграждение, но также влияет на вознаграждение на всех последующих временных шагах. Например, используя соответствующие методы обучения с подкреплением, нейронная сеть может научиться играть в нарды на высоком уровне. Здесь сеть должна научиться принимать позицию доски в качестве входных данных вместе с результатом броска костей и производить сильный ход в качестве выходных данных. Это достигается тем, что сеть играет против своей копии, возможно, за миллион игр. Основная проблема заключается в том, что игра в нарды может включать в себя десятки ходов, и все же только в конце игры достигается награда в форме победы. Затем награда должна быть соответствующим образом отнесена ко всем ходам, которые привели к ней, даже если некоторые ходы были хорошими, а другие - не такими. Это пример проблемы с присвоением кредита. Общей особенностью обучения с подкреплением является компромисс между исследованием, в котором система пробует новые виды действий, чтобы увидеть, насколько они эффективны, и эксплуатацией, в которой система использует действия, которые, как известно, дают высокую награду. Слишком сильный акцент на разведку или эксплуатацию приведет к плохим результатам. Усиленное обучение продолжает оставаться активной областью исследований машинного обучения.

Все это можно изучить, представив График набора обучающих данных с $N = 10$ баллами, показанный в виде синих кружков, каждый из которых содержит наблюдение входной переменной $x$ вместе с соответствующей целевой переменной $t$. Зеленая кривая показывает функцию $\sin(2\pi x)$, используемую для генерации данных. Наша цель - предсказать значение $t$ для некоторого нового значения $x$ без знания зеленой кривой (рисунок 1.20).

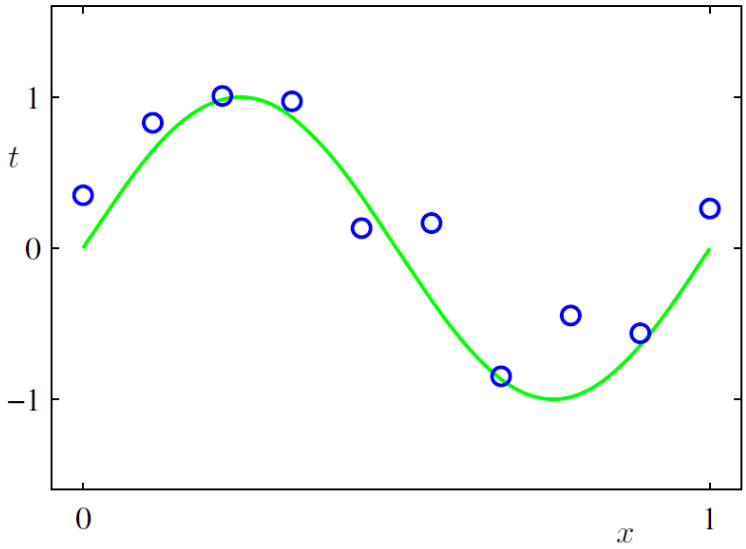


Рисунок 1.20 - График набора тренировочных данных

Ключевой концепцией в области распознавания образов является понятие неопределенности. Он возникает как из-за шума при измерениях, так и из-за конечного размера наборов данных. Теория вероятностей обеспечивает непротиворечивую основу для количественного определения и манипулирования неопределенностью и формирует одну из центральных основ для распознавания образов. В сочетании с теорией принятия решений, это позволяет нам делать оптимальные прогнозы, учитывая всю доступную нам информацию, даже если эта информация может быть неполной или неоднозначной.

Мы введем основные понятия теории вероятностей на простом примере. Представьте, что у нас есть две коробки, одна красная и одна синяя, в красной коробке 2 яблока и 6 апельсинов, а в синей коробке 3 яблока и 1 апельсин.

Это показано на рисунке 1.20, Теперь предположим, что мы случайным образом выбираем одну из коробок, и из этой коробки мы случайным образом выбираем фрукт, и, заметив, какой это фрукт, мы заменяем его в ящике, из которого он получен. Мы могли бы представить, что повторяем этот процесс много раз. Предположим, что при этом мы выбираем красную коробку в 40% случаев, и мы выбираем синюю коробку в 60% случаев, и что при удалении фруктового ящика из коробки мы с равной вероятностью выберем любую из кусочков фруктов в коробке.

В этом примере идентификатор ячейки, которая будет выбрана, является случайной величиной, которую мы обозначим через B. Эта случайная переменная может принимать одно из двух возможных значений, а именно r (соответствует красному прямоугольнику) или b (соответствует синяя коробка). Аналогично, идентичность плода также является случайной величиной и будет обозначаться F. Она может принимать одно из значений a (для яблока) или o (для апельсина). Для начала, мы определим вероятность события как долю случаев, когда это событие происходит из общего числа

испытаний, в пределе, когда общее количество испытаний переходит в бесконечность (рисунок 1.21).

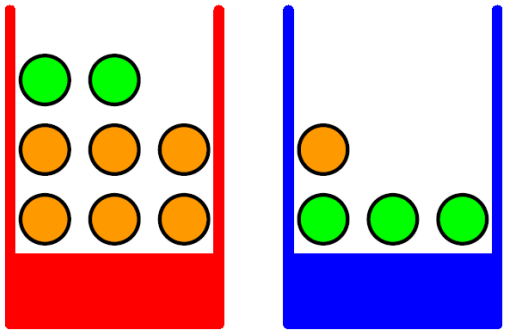

Рисунок 1.21 – Пример для изучения понятия неопределенности

Таким образом, вероятность выбора красного ящика составляет 4/10, а вероятность выбора синего ящика - 6/10. Запишем эти вероятности как p(B = r) = 4/10 и p (B = b) = 6/10. Отметим, что по определению вероятности должны лежать в интервале [0, 1]. Кроме того, если события являются взаимоисключающими и если они включают в себя все возможные результаты (например, в этом примере поле должно быть либо красного, либо синего цвета), то мы видим, что вероятности для этих событий должны быть равны единице. Теперь мы можем задавать вопросы, такие как: «какова общая вероятность того, что процедура отбора выберет яблоко» Или «учитывая, что мы выбрали апельсин, какова вероятность того, что выбранная нами коробка была синей», Мы можем ответить на такие вопросы и даже на гораздо более сложные вопросы, связанные с проблемами распознавания образов, как только мы оснастим себя двумя простейшими правилами вероятности, известными как правило сумм и правило произведений. Получив эти правила, мы вернемся к нашему примеру с фруктами (рисунок 1.22).

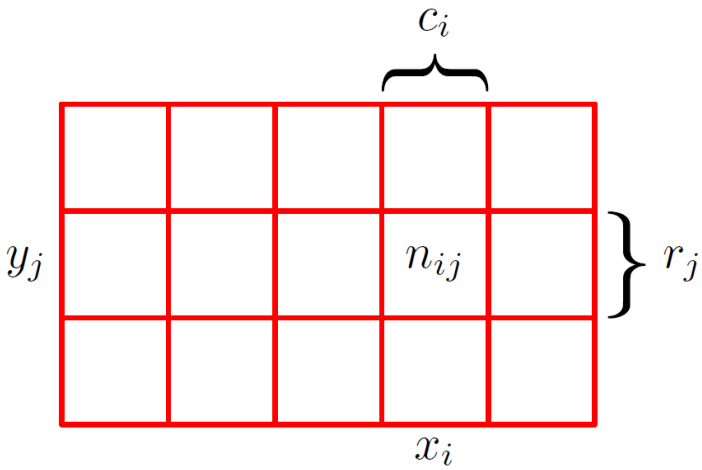


Рисунок 1.22 – Сумма и произведение вероятностных правил

Чтобы вывести правила вероятности, рассмотрим несколько более общий пример, показанный на рисунке 1.21, в котором участвуют две

случайные величины X и Y (например, это могут быть переменные Бокса и Фрукта, рассмотренные выше). Предположим, что X может принимать любое из значений $x_i$, где i = 1,…, M и Y могут принимать значения $y_j$, где j = 1, … , L. Рассмотрим всего N испытаний, в которых мы отобрали обе переменные X и Y, и пусть количество таких испытаний, в которых $X = x_i$ и $Y = y_j$, равно $n_{ij}$. Кроме того, пусть число испытаний, в которых X принимает значение $x_i$ (независимо от значения, которое принимает Y), будет обозначаться через $c_i$, и аналогично пусть число испытаний, в которых Y принимает значение $y_j$, будет обозначаться через $r_j$.

$$p(X = x_i, Y = y_i) = \frac{n_{ij}}{N} \tag{1.10}$$

Здесь мы неявно рассматриваем предел N → ∞. Точно так же вероятность того, что X принимает значение $x_i$ независимо от значения Y, записывается как p ($X = x_i$) и определяется как доля общего числа точек, попадающих в столбец i, так что:

$$p(X = x_i) = \frac{c_i}{N} \tag{1.11}$$

Поскольку количество экземпляров в столбце i на рисунке 21 является просто суммой количества экземпляров в каждой ячейке этого столбца, у нас имеется $c_i = \sum n_{ij}$ и поэтому, исходя из (1.10) и (1.11):

$$p(X = x_i) = \sum_{j=1}^{L} p(X = x_i\,, Y = y_i) \tag{1.12}$$

В итоге, у нас есть уравнение, которое является правилом суммы вероятностей. Обратите внимание, что p ($X = x_i$) иногда называют предельной вероятностью, потому что она получается путем маргинализации или суммирования других переменных (в данном случае Y).

Если мы рассмотрим только те случаи, для которых $X = x_i$, то доля таких случаев, для которых $Y = y_j$, записывается как p ($Y = y_j \mid X = x_i$) и называется условной вероятностью $Y = y_j$ при $X = x_i$. Он получается путем нахождения доли тех точек в столбце i, которые попадают в ячейку i, j и, следовательно, определяется как:

$$p(Y = y_i, X = x_i) = \frac{n_{ij}}{c_i} \tag{1.13}$$

Из (1.10), (1.11) и (1.11) можно получить следующее соотношение, которое является правилом вероятности произведения:

$$p(Y = y_i, X = x_i) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} * \frac{c_i}{N} = p(Y = y_i, X = x_i)p(X = x_i) \quad (1.14)$$

До сих пор мы довольно тщательно проводили различие между случайной переменной, такой как блок B в примере с фруктами, и значениями, которые может принимать случайная переменная, например, r, если блок был красным. Таким образом, вероятность того, что B принимает значение r, обозначается как p (B = r). Хотя это помогает избежать двусмысленности, оно приводит к довольно громоздким обозначениям, и во многих случаях такая педантичность не требуется. Вместо этого мы можем просто написать p (B), чтобы обозначить распределение по случайной переменной B, или p (r), чтобы обозначить распределение, оцененное для конкретного значения r, при условии, что интерпретация ясна из контекста.

С помощью этого более компактного обозначения мы можем записать два основных правила теории вероятностей в следующем виде:

$$\text{правило сумм:} \qquad p(X) = \sum_{Y} p(X, Y) \quad (1.15)$$

$$\text{правило итога:} \qquad p(X, Y) = p(Y, X)p(X) \quad (1.16)$$

Как правило, в изучении вопроса создания и разработки систем, использующих компьютерное зрение, нельзя не изучить регуляризацию алгоритмов обучения.

Согласно Гудфеллоу и др., Регуляризация — это «любая модификация алгоритма обучения, которую мы вносим, чтобы уменьшить ошибку его обобщения, но не ошибку обучения». Короче говоря, регуляризация стремится уменьшить нашу ошибку тестирования, возможно, за счет небольшого увеличения ошибки обучения.

Однако параметризованные формы регуляризации, которые требовали от нас обновления функции потери / обновления. На самом деле существуют другие типы регуляризации, которые либо:

1. Изменить саму сетевую архитектуру.
2. Дополнить данные, передаваемые в сеть для обучения.

Метод исключения является отличным примером изменения сетевой архитектуры путем достижения большей обобщения. Здесь мы вставляем слой, который случайным образом отключает узлы от предыдущего уровня до следующего уровня, тем самым гарантируя, что ни один узел не отвечает за обучение представлению данного класса.

Также, мы обсудим другой тип регуляризации, называемый дополнением данных. Этот метод преднамеренно мешает обучающим примерам, слегка изменяя их внешний вид, прежде чем передавать их в сеть для обучения. Конечным результатом является то, что сеть постоянно видит «новые» точки обучающих данных, сгенерированные из исходных

обучающих данных, что частично устраняет необходимость сбора дополнительных обучающих данных (хотя в целом сбор дополнительных обучающих данных редко повредит вашему алгоритму).

Увеличение данных включает в себя широкий спектр методов, используемых для генерации новых обучающих выборок из исходных с применением случайных дрожаний и возмущений, так что метки классов не изменяются. Наша цель при применении дополнения данных - повысить обобщаемость модели. Учитывая, что в нашей сети постоянно появляются новые, слегка измененные версии точек входных данных, она может изучать более надежные функции. Во время тестирования мы не применяем расширение данных и не оцениваем нашу обученную сеть - в большинстве случаев вы увидите увеличение точности тестирования, возможно, за счет небольшого снижения точности обучения. Рассмотрим некоторую выборку данных нормального распределения (рисунок 1.23), справа: добавление небольшого количества случайного «джиттера» в дистрибутив. Этот тип увеличения данных может увеличить обобщаемость наших сетей.

Давайте рассмотрим рисунок 1.23 (слева) нормального распределения с нулевым средним и единичной дисперсией. Обучение модели машинного обучения на этих данных может привести к точному моделированию распределения - однако в реальных приложениях данные редко следуют такому аккуратному распределению.

Вместо этого, чтобы повысить обобщаемость нашего классификатора, мы можем сначала произвольно дрожать точки вдоль распределения, добавляя некоторые значения, взятые из случайного распределения (справа). Наш график все еще соответствует примерно нормальному распределению, но это не идеальное распределение, как слева. Модель, обученная на этих данных, с большей вероятностью обобщает примерные точки данных, не включенные в обучающий набор.

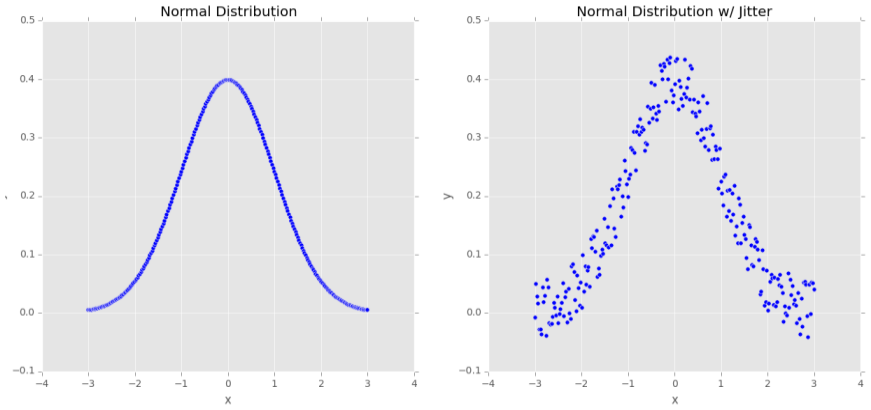


Рисунок 1.23 - Выборка из 250 точек данных, которые точно следуют нормальному распределению

В контексте компьютерного зрения расширение данных само собой разумеющееся. Например, мы можем получить дополнительные обучающие

данные из исходных изображений, применив простые геометрические преобразования, такие как случайные:

1. переводы;
2. вращения;
3. изменения в масштабе;
4. сдвиги;
5. горизонтальные (а в некоторых случаях вертикальные) отражения.

Применение (небольшого) количества этих преобразований к входному изображению немного изменит его внешний вид, но не изменит метку класса - тем самым сделав увеличение данных очень естественным и простым методом. применять для глубокого обучения для задач компьютерного зрения. Более продвинутые методы увеличения данных, применяемые к компьютерному зрению, включают случайное возмущение цветов в данном цветовом пространстве и нелинейные геометрические искажения. Представим случайное изображение, с некоторыми отличиями, для пополнения базы данных нейронной сети (рисунок 1.24).

Рисунок 1.24 - Входное изображение, к которому применяется расширение данных.

Был произведен монтаж каждого из этих изображений, чтобы была возможность визуализировать их на рисунке 1.24 (справа). Обратите внимание, как каждое изображение было случайно повернуто, срезано, масштабировано и перевернуто по горизонтали. В каждом случае изображение сохраняет оригинальную метку класса: собака; тем не менее, каждое изображение было немного изменено, что дало нашей нейронной сети новые шаблоны для изучения при обучении. Поскольку входные изображения будут постоянно меняться (хотя метки классов остаются неизменными), мы часто наблюдаем снижение точности обучения по сравнению с тренировками без увеличения данных. Однако, как мы узнаем позже в этой главе, расширение данных может помочь значительно сократить переоснащение, в то же время гарантируя, что наша модель лучше обобщается для новых входных выборок. Кроме того, при работе с наборами данных, где у нас слишком мало примеров для применения глубокого обучения, мы можем использовать увеличение данных для генерации дополнительных обучающих данных, тем самым уменьшая количество данных, помеченных вручную, необходимых для обучения сети глубокого обучения.

## 2 Конструкторская часть

### 2.1 Микрокомпьютер как основа системы

Базовые блоки микрокомпьютера. Микрокомпьютер имеет три основных блока: центральный процессор (ЦП), блок памяти и блок ввода-вывода. Процессор выполняет все инструкции и выполняет арифметические и логические операции с данными. Процессор микрокомпьютера называется «микропроцессор». Микропроцессор, как правило, представляет собой одну микросхему VLSI (очень большая интеграция), которая содержит все регистры, блок управления и арифметические / логические схемы микрокомпьютера. Блок памяти хранит как данные, так и инструкции. Раздел памяти обычно содержит микросхемы ПЗУ и ОЗУ. ПЗУ доступно только для чтения и является энергонезависимым, то есть сохраняет свое содержимое при выключении питания. ПЗУ обычно используется для хранения инструкций и данных, которые не меняются. Например, он может хранить таблицу кодов для вывода данных на дисплей, внешний по отношению к микрокомпьютеру, для включения цифры от 0 до 9. Можно читать и записывать в RAM. Оперативная память нестабильна; то есть он не сохраняет свое содержимое при выключении питания. ОЗУ используется для хранения программ и данных, которые являются временными и могут измениться в ходе выполнения программы. Модуль 110 (Input/Output) передает данные между микрокомпьютером и внешними устройствами через порты ввода / вывода (регистры). Передача включает в себя данные, статус и сигналы управления. В однокристальном микрокомпьютере эти три элемента находятся на одном кристалле, тогда как для одночипового микропроцессора требуются отдельные микросхемы для памяти и ввода / вывода (рисунок 2.1).

Микроконтроллеры развивались из однокристальных микрокомпьютеров. Микроконтроллеры обычно используются для специальных приложений, таких как автомобильные системы, бытовая техника и домашние развлекательные системы. Поэтому типичные микроконтроллеры включают встроенные таймеры и аналого-цифровые (аналогово-цифровые) и цифро-аналоговые (цифро-аналоговые) преобразователи (рисунок 2.2).

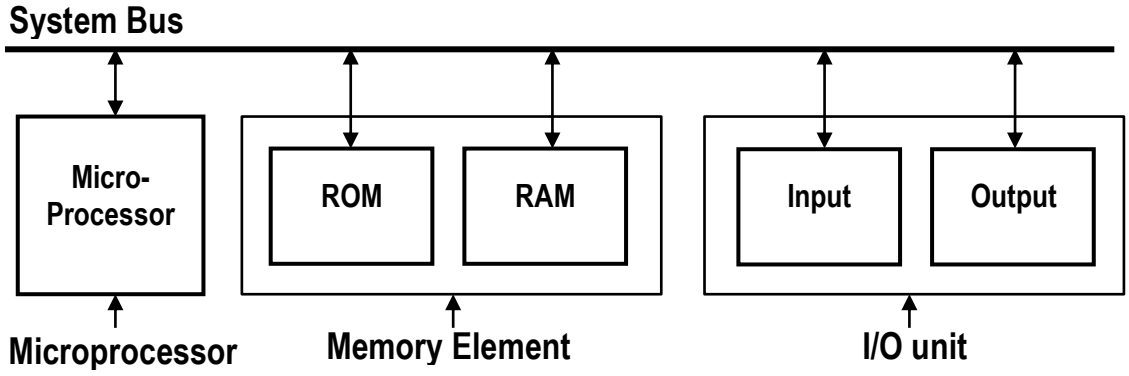

Рисунок 2.1 - Основные блоки микрокомпьютера

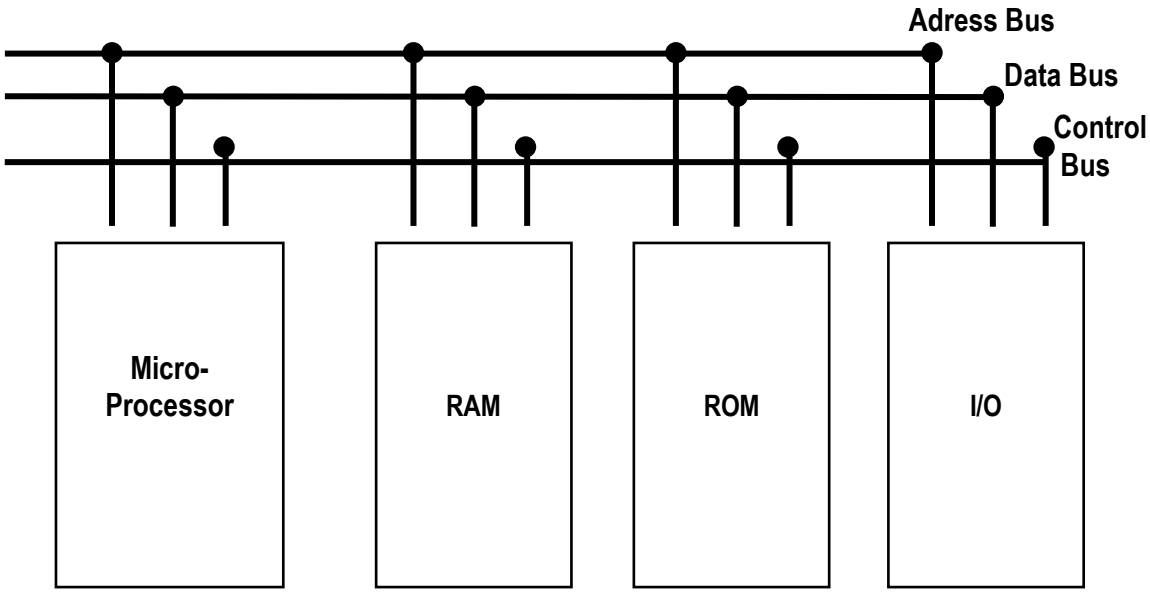


Рисунок 2.2 - Упрощенная версия типичного микрокомпьютера

Микроконтроллерами являются Intel 875 1 (8 бит) / 8096 (16 бит) и Motorola HC11 (8 бит) / HC16 (16 бит). 16-битные микроконтроллеры содержат больше встроенного ПЗУ, ОЗУ и ввода-вывода, чем микроконтроллеры% бит. На рисунке 2.2 показаны основные блоки микрокомпьютера. Системная шина (состоящая из нескольких проводов) соединяет эти блоки.

Архитектура стандартного микрокомпьютера Различные микрокомпьютеры, доступные сегодня, в принципе одинаковы. Основные изменения заключаются в количестве битов данных и адресов и в типах сигналов управления, которые они используют.

Чтобы понять основные принципы архитектуры микрокомпьютера, необходимо детально исследовать типичный микрокомпьютер. Как только будет получено такое четкое понимание, станет легче работать с любым конкретным микрокомпьютером. Рисунок 2.2 иллюстрирует наиболее упрощенную версию типичного микрокомпьютера. На рисунке показаны основные блоки микрокомпьютерной системы. Различные шины, которые соединяют эти блоки, также показаны. Хотя этот рисунок выглядит очень просто, он включает в себя все основные элементы типичной микрокомпьютерной системы.

Микрокомпьютерная шина. Системная шина микрокомпьютера содержит три шины, которые несут всю адресную информацию, данные и управляющую информацию, связанную с выполнением программы. Эти шины соединяют микропроцессор (ЦП) с каждым из ПЗУ, ОЗУ и микросхем ввода-вывода, чтобы можно было осуществлять обмен информацией между микропроцессором и любыми другими элементами.

В микрокомпьютере типичная передача информации осуществляется в отношении памяти или ввода-вывода. Когда память или микросхема ввода-вывода получает данные от микропроцессора, это называется операцией WRITE, и данные записываются в выбранную ячейку памяти или порт 110 (регистр). Когда память или микросхема ввода-вывода отправляют данные в микропроцессор, это называется операцией READ, и данные считываются из выбранной ячейки памяти или порта ввода-вывода.

В адресной шине передача информации происходит только в одном направлении, от микропроцессора к памяти или 110 элементам. Поэтому это называется «однонаправленная шина». Длина этой шины обычно составляет от 20 до 32 бит. Размер адресной шины определяет общее количество доступных адресов памяти, в которых программы могут выполняться микропроцессором. Адресная шина определяется общим количеством адресных контактов на микропроцессорной микросхеме. Это также определяет возможность прямой адресации или размер основной памяти микропроцессора. Микропроцессор может выполнять только те программы, которые находятся в основной памяти.

Например, микропроцессор с 20 адресными контактами может генерировать $2^{20}$= 1 048 576 (один мегабайт) различных возможных адресов (комбинаций 1 и 0) на адресной шине. Микропроцессор содержит адреса от 0 до 1 048 575 ($OOOOO_{16}$ через $FFFF_{16}$). Место в памяти может быть представлено каждым из этих адресов. Например, 8-битный элемент данных может быть сохранен по адресу $00200_{16}$. Когда микропроцессор, такой как 8086, хочет передать информацию между собой и определенной ячейкой памяти, он генерирует 20-битный адрес из внутреннего регистра на своих 20 адресных выводах $A_0$ – $A_{19}$ который затем появляется на адресной шине , Эти 20 адресных битов декодируются для определения желаемой ячейки памяти. Процесс декодирования обычно требует аппаратных средств (декодеров), не показанных на рисунке 2.2.

В шине данных данные могут передаваться в обоих направлениях. Следовательно, это двунаправленная шина. В некоторых микропроцессорах контакты данных используются для отправки другой информации, такой как биты адреса, в дополнение к данным. Это означает, что выводы данных используются совместно или по времени. Микропроцессор Intel 8086 является примером, в котором 20-битный адрес мультиплексируется с 16-битной шиной данных и четырьмя линиями состояния.

Шина управления состоит из ряда сигналов, которые используются для синхронизации работы отдельных элементов микрокомпьютера. Микропроцессор отправляет некоторые из этих управляющих сигналов другим элементам, чтобы указать тип выполняемой операции. Каждый микрокомпьютер имеет уникальный набор управляющих сигналов. Однако есть некоторые управляющие сигналы, которые являются общими для большинства микропроцессоров. Мы опишем некоторые из этих управляющих сигналов позже в этом разделе.

Тактовые сигналы. Системные тактовые сигналы содержатся в шине управления. Эти сигналы генерируют соответствующие такты, в течение которых микропроцессор выполняет инструкции. Тактовые сигналы варьируются от одного микропроцессора к другому. Некоторые микропроцессоры имеют внутреннюю схему тактового генератора для генерации тактового сигнала. Эти микропроцессоры требуют подключения внешнего кристалла или RC-сети к соответствующим контактам микропроцессора для установки рабочей частоты. Например, для Intel 801 86 (16-разрядный микропроцессор) не требуется схема внешнего тактового генератора. Однако большинство микропроцессоров не имеют схемы внутреннего тактового генератора и требуют внешнего чипа или схемы для генерации тактового сигнала. Рисунок 2.3 показывает типичный тактовый сигнал. Одночипный микропроцессор. Как упоминалось ранее, микропроцессор является процессором микрокомпьютера. Следовательно, мощность микрокомпьютера определяется возможностями микропроцессора. Его тактовая частота определяет скорость работы микрокомпьютера. Количество контактов данных и адреса на микропроцессорной микросхеме составляют размер слова и максимальный объем памяти микрокомпьютера. Возможности ввода-вывода и сопряжения микрокомпьютера определяются управляющими выводами микропроцессорной микросхемы.

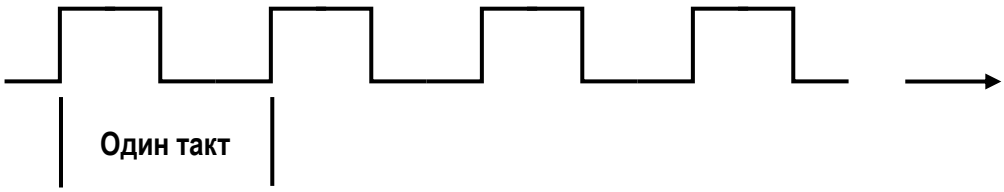


Рисунок 2.3 - Типичный тактовый сигнал

| |
|---|
| Registers |
| ALU |
| Control Unit |

Рисунок 2.4 - Микропроцессорная микросхема с основными функциональными элементами

Логика внутри микропроцессорной микросхемы может быть разделена на три основные области: секция регистров, блок управления и арифметико-логический блок (АЛУ). Микропроцессорный чип с этими тремя секциями показан на рисунке 2.4. Теперь опишем эти разделы.

Регистры. Количество, размер и типы регистров варьируются от одного микропроцессора к другому. Однако различные регистры во всех микропроцессорах выполняют аналогичные операции. Структуры регистров

микропроцессоров играют важную роль в проектировании микропроцессорных архитектур. Кроме того, структуры регистров для конкретного микропроцессора определяют, насколько удобно и легко программировать этот микропроцессор. Сначала мы опишем самые основные типы микропроцессорных регистров, их функции и способы их использования. Затем мы рассмотрим другие распространенные типы регистров. Основные микропроцессорные регистры. Существует четыре основных микропроцессорных регистра: регистр команд, счетчик программ, регистр адресов памяти и аккумулятор.

1) Регистр команд (РК). Регистр команд хранит инструкции. Содержимое регистра команд всегда декодируется микропроцессором как инструкция. После извлечения кода инструкции из памяти микропроцессор сохраняет его в регистре команд. Инструкция декодируется внутренне микропроцессором, который затем выполняет требуемую операцию. Размер слова микропроцессора определяет размер регистра команд. Например, 16-разрядный микропроцессор имеет 16-разрядный регистр команд.

2) счетчик программ (ПК). Счетчик программы содержит адрес инструкции или код операции (код операции). Программный счетчик обычно содержит адрес следующей команды, которая должна быть выполнена. Обратите внимание на следующие особенности счетчика программ:

а) после активации входа RESET микропроцессора адрес первой команды, которая должна быть выполнена, загружается в счетчик программы.

б) для выполнения инструкции микропроцессор обычно помещает содержимое счетчика программы на адресную шину и считывает («выбирает») содержимое этого адреса, то есть инструкцию, из памяти. Содержимое счетчика программ автоматически увеличивается с помощью внутренней логики микропроцессора. Таким образом, микропроцессор выполняет программу последовательно, если только программа не содержит инструкцию, такую как инструкция JUMP, которая изменяет последовательность.

в) размер счетчика программ определяется размером адресной шины.

г) многие инструкции, такие как JUMP и условный JUMP, изменяют содержимое счетчика программы с обычного значения последовательного адреса. Счетчик программы загружается с адреса, указанного в этих инструкциях.

3) регистр адресов памяти (MAR). Регистр адреса памяти содержит адрес данных. Микропроцессор использует адрес, который хранится в регистре адресов памяти, в качестве прямого указателя на память. Содержимое адреса состоит из фактических данных, которые передаются.

4) аккумулятор (A). Для 8-разрядного микропроцессора аккумулятор обычно представляет собой 8-разрядный регистр. Он используется для хранения результата после большинства операций ALU. Эти микропроцессоры имеют инструкции сдвинуть или повернуть аккумулятор на 1 бит вправо или влево через флаг переноса. Аккумулятор обычно

используется для ввода байта в аккумулятор от внешнего устройства или для вывода байта на внешнее устройство из аккумулятора.

Некоторые микропроцессоры, такие как Motorola 6809, имеют более одного аккумулятора. В этих микропроцессорах аккумулятор, используемый командой, указан в коде операции.

В зависимости от секции регистров микропроцессор может быть классифицирован как машина на основе аккумулятора или машина общего назначения. В микропроцессоре на основе аккумулятора, таком как Intel 8085 и Motorola 6809, предполагается, что данные хранятся в регистре, называемом «аккумулятор». Все арифметические и логические операции выполняются с использованием этого регистра в качестве одного из источников данных. Результат после операции сохраняется в аккумуляторе. Восьмибитные микропроцессоры обычно основаны на аккумуляторе.

В зависимости от секции регистров микропроцессор может быть классифицирован как машина на основе аккумулятора или машина общего назначения. В микропроцессоре на основе аккумулятора, таком как Intel 8085 и Motorola 6809, предполагается, что данные хранятся в регистре, называемом «аккумулятор». Все арифметические и логические операции выполняются с использованием этого регистра в качестве одного из источников данных. Результат после операции сохраняется в аккумуляторе. Восьмибитные микропроцессоры обычно основаны на аккумуляторе.

Использование базовых микропроцессорных регистров. Чтобы обеспечить четкое понимание того, как используются основные микропроцессорные регистры, будет рассмотрена программа двоичного сложения. Логика программы будет объяснена показом того, как каждая инструкция изменяет содержимое четырех регистров. Предположим, что все числа в шестнадцатеричном виде. Предположим, что содержимое ячейки памяти 2018 нужно добавить к содержимому 2020 года. Предположим, что [NNNN] представляет содержимое памяти.

Теперь предположим, что [2018] = 0002 и [2020] = 0005. Шаги, необходимые для выполнения этого добавления, можно суммировать следующим образом:

1) загрузить регистр адреса памяти (MAR) с адресом первого добавляемого элемента данных, то есть загрузите 2018 в MAR.

2) переместить содержимое этого адреса в регистр данных, DO; переместите первые данные в DO.

3) увеличить значение MAR на 2, чтобы сохранить 2020, адрес второго элемента данных, который будет добавлен.

4) добавить содержимое этой ячейки памяти к данным, которые были перемещены в регистр данных, DO на шаге 2, и сохраните результат в 16-битном регистре данных, DO. Вышеуказанная дополнительная программа будет написана с использованием 68000 инструкций. Обратите внимание, что 68000 использует 24-битные адреса; 24-битные адреса, такие как $002000_{16}$ будут представлены как $002000_{16}$ (1 6-битное число) в следующем.

а) загрузить содержимое следующего 16-битного слова памяти в регистр адреса памяти, Al. Обратите внимание, что регистр A1 можно считать MAR в 68000.

б) считать 16-битное содержимое ячейки памяти, адресуемой MAR, в регистр данных DO.

в) увеличить значение MAR на 2, чтобы сохранить 2020, адрес вторых данных, которые будут добавлены.

г) добавить текущее содержимое регистра данных, DO к содержимому ячейки памяти, адрес которой находится в MAR, и сохраните 16-битный результат в DO.

Следующие шаги для Motorola 68000 будут использоваться для достижения вышеуказанного дополнения:

$3279_{16}$ загрузить содержимое следующего 1 6-битного слова памяти в память адресный регистр.

$3010_{16}$ Считать 16-битное содержимое ячейки памяти, адресуемой MAR, в регистр данных, DO.

$5249_{16}$ увеличение MAR на 2.

$DO51_{16}$ добавить текущее содержимое регистра данных, DO, к содержимому ячейки памяти, адрес которой находится в MAR, и сохранить 16-битный результат в DO.

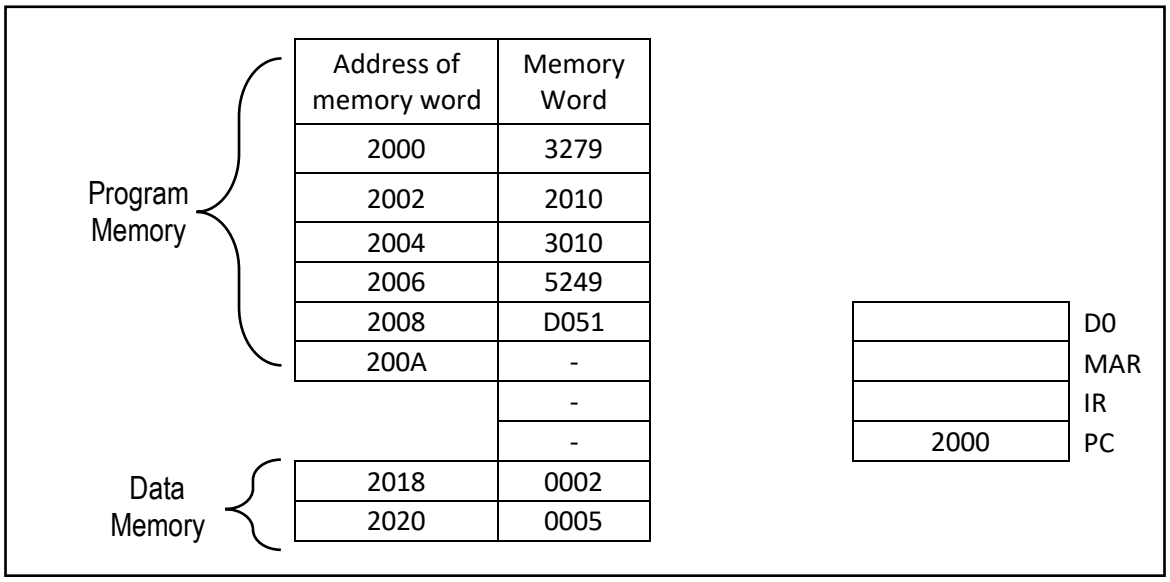


Рисунок 2.5 - Программа добавления микропроцессора с начальным регистром и памятью

Полная программа в шестнадцатеричном формате, начиная с местоположения $2000_{16}$ (произвольно выбранного), приведена на рисунке 2.5 Обратите внимание, что каждый адрес памяти хранит 16 бит. Следовательно, адреса памяти показаны с шагом 2. Предположим, что микрокомпьютер может быть проинструктирован, что начальный адрес программы $2000_{16}$. Это означает, что счетчик программ может быть инициализирован, чтобы содержать $2000_{16}$ адрес первой команды, которая должна быть выполнена.

Обратите внимание, что содержимое трех других регистров на данный момент неизвестно. Микропроцессор загружает содержимое ячейки памяти, адресуемой счетчиком программы, в ИК-порт. Таким образом, первая инструкция, $3279_{16}$ хранящаяся в адресе $2000_{16}$ передается в IR.

Далее, разберем архитектуру микрокомпьютера Raspberry Pi 3 B+, который был выбран в качестве основного процессора системы распознавания.

Еще в 2006 году, когда Эбен Аптон и его коллеги из Кембриджского университета совместно с Питом Ломасом и Дэвидом Брабеном создали Фонд Raspberry Pi.

Ранние прототипы Raspberry Pi были основаны на 8-битном Atmel ATMega644, чтобы снизить стоимость. В следующих прототипах использовался процессор ARM, аналогичный тому, который использовался в релизной версии Raspberry Pi.

В 2012 году команда начала свой первый производственный цикл, состоящий из 10 000 установок Raspberry Pi, изготовленных литейными заводами в Китае и на Тайване.

Графический процессор обеспечивает Open GL ES 2.0, аппаратное ускорение Open VG и высокопрофильное декодирование 1080p30 H.264 и способен вычислять 1 Gpixel / s, 1.5Gtexel / s или 24 GFLOP общего назначения. Это означает, что если вы подключите Raspberry Pi 3 к вашему HDTV, вы сможете смотреть видео с качеством BluRay, используя H.264 со скоростью 40 Мбит / с (рисунок 2.6).

Самое большое изменение, которое было введено с Raspberry Pi 3, - это обновление до основного процессора следующего поколения и улучшенная связь с Bluetooth Low Energy (BLE) и BCM43143 Wi-Fi на борту. Кроме того, Raspberry Pi 3 имеет улучшенное управление питанием с обновленным коммутируемым источником питания до 2,5 A для поддержки более мощных внешних USB-устройств (рисунок 2.6).

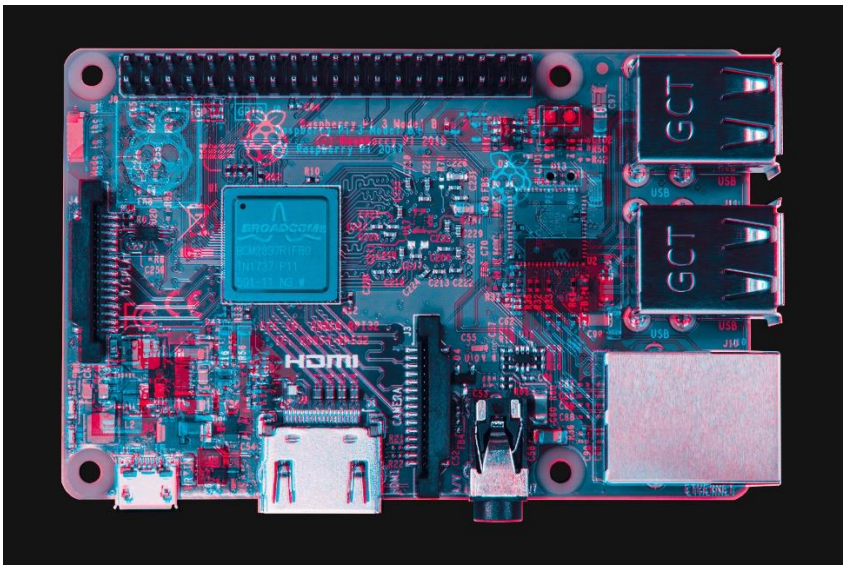

Рисунок 2.6 – Изображение Raspberry Pi 3B+

Четыре встроенных USB-порта Raspberry Pi 3 обеспечивают достаточное количество подключений для мыши, клавиатуры или чего-либо еще, что вам нужно для RPi, но если вы хотите добавить еще больше, вы все равно можете использовать концентратор USB. Имейте в виду, что рекомендуется использовать концентратор с питанием, чтобы не перегружать встроенный регулятор напряжения. Подключить Raspberry Pi 3 легко, просто подключите любой USB-блок питания к порту micro-USB. Там нет кнопки питания, поэтому Pi начнет загружаться, как только будет подано питание, чтобы отключить его, просто отключите питание. Четыре встроенных USB-порта могут выдавать до 1,2 А, что позволяет подключать более энергоемкие USB-устройства (для этого требуется блок питания Micro USB 2 А).

Процессор SoC. Созданная специально для нового Pi 3, система на кристалле Broadcom BCM2837 включает в себя четыре высокопроизводительных процессора ARM Cortex-A53, работающих на частоте 1,2 ГГц с кэш-памятью уровня 1 32 КБ и 512 КБ уровня 2, графический процессор VideoCore IV и подключен к модулю памяти LPDDR2 емкостью 1 ГБ на задней панели (рисунок 2.7).

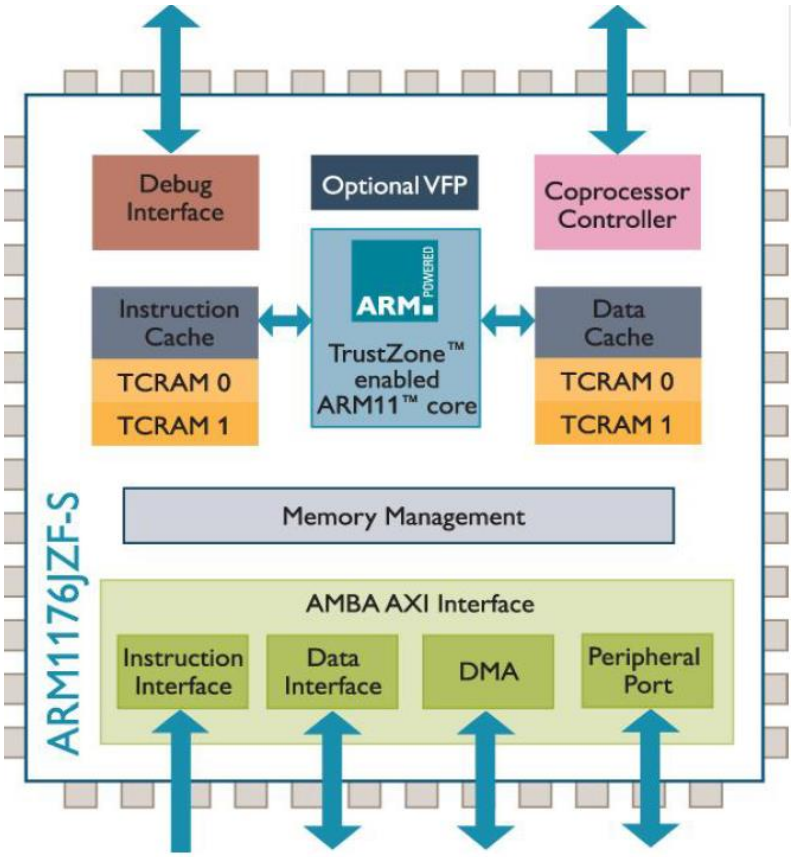


Рисунок 2.7 – Архитектура процессора

Характеристики, выделяющие данную модель от других:

-SoCNext Generation QUAD Core Broadcom BCM2837 64-битный процессор ARMv7;

- Скорость процессора увеличилась с 900 МГц на Pi 2 до 1,25 ГГц на RPi 3 Model B;

- BCM43143 Wi-Fi на борту;

- Bluetooth Low Energy (BLE) на борту;

- Модернизированный коммутируемый источник питания до 2,5 А (теперь можно подключать даже более мощные устройства через порты USB). Основными отличиями являются четырехъядерный 64-битный процессор и встроенные Wi-Fi и Bluetooth. Объем оперативной памяти остается 1 ГБ, и нет изменений в портах USB или Ethernet. Однако улучшенное управление питанием должно означать, что Pi 3 может использовать более энергоемкие USB-устройства. Для Raspberry Pi 3 Broadcom поддержал нас с новым SoC, BCM2837. При этом сохраняется та же базовая архитектура, что и у его предшественников BCM2835 и BCM2836, поэтому все те проекты и учебные пособия, которые основаны на точных деталях оборудования Raspberry Pi, будут продолжать работать. 32-битный четырехъядерный процессор ARM CortexA7 с частотой 900 МГц был заменен 64-битным четырехъядерным процессором ARM Cortex-A53 с частотой 1,2 ГГц. По размерам он идентичен B + и Pi 2. Все разъемы и монтажные отверстия находятся в одном месте, поэтому все существующие надстройки, HAT и корпуса должны подходить, хотя светодиоды питания и активности переместились, чтобы освободить место для антенны WiFi. Производительность Pi 3 примерно на 50-60% выше, чем у Pi 2, что означает, что он в десять раз быстрее, чем у исходного Pi.

Созданная специально для нового Pi 3, система на кристалле Broadcom BCM2837 включает в себя четыре высокопроизводительных процессора ARM Cortex-A53, работающих на частоте 1,2 ГГц с кэш-памятью уровня 1 32 кБ и 512 кБ уровня 2, графический процессор VideoCore IV и подключен к модулю памяти LPDDR2 емкостью 1 ГБ на задней панели (рисунок 2.8). Саму же принципиальную электрическую схему микрокомпьютера можно рассмотреть в приложении 1.

| | Raspberry Pi 3 Model B | Raspberry Pi 2 Model B | Model B+ | Model A+ | Model A | CMDK |
|---|---|---|---|---|---|---|
| **Processor Chipset** | Broadcom BCM2837 64Bit ARMv7 Quad Core Processor powered Single Board Computer running at 1250MHz | Broadcom BCM2836 32bit ARMv7 Quad Core Processor powered Single Board Computer running at 900MHz | Broadcom BCM2835 32bit ARMv6 SoC full HD multimedia applications processor | Broadcom BCM2835 32bit ARMv6 SoC full HD multimedia applications processor | Broadcom BCM2835 32bit ARMv6 SoC full HD multimedia applications processor | Broadcom BCM2835 32bit ARMv6 SoC full HD multimedia applications processor |
| **GPU** | Videocore IV | Videocore IV | Videocore IV | Videocore IV | Videocore IV | Videocore IV |
| **Processor Speed** | QUAD Core @1250 MHz | QUAD Core @900 MHz | Single Core @700 MHz | Single Core @700 MHz | Single Core @700 MHz | Single Core @700 MHz |
| **RAM** | 1GB SDRAM @ 400 MHz | 1GB SDRAM @ 400 MHz | 512 MB SDRAM @ 400 MHz | 256 MB SDRAM @ 400 MHz | 256 MB SDRAM @ 400 MHz | 512 MB SDRAM @ 400 MHz |
| **Storage** | MicroSD | MicroSD | MicroSD | MicroSD | SDCard | 4GB eMMC |
| **USB 2.0** | 4x USB Ports | 4x USB Ports | 4x USB Ports | 1x USB Port | 1x USB Port | 1x USB Port |
| **Power Draw / voltage** | 2.5A @ 5V | 1.8A @ 5V | 1.8A @ 5V | 1.8A @ 5V | 1.2A @ 5V | 1.8A @ 5V |
| **GPIO** | 40 pin | 40 pin | 40 pin | 40 pin | 26 pin | 120 pin |
| **Ethernet Port** | Yes | Yes | Yes | No | No | No |
| **Wi-Fi** | Built in | No | No | No | No | No |
| **Bluetooth LE** | Built in | No | No | No | No | No |

Рисунок 2.8 – Сравнительная таблица моделей микрокомпьютера

## 2.2 Камера как основной элемент считывания данных

В разработке данного проекта, в качестве главного элемента считывания данных была выбрана Raspberry Pi – совместимая широкоугольная камера Fish eye, комплектация камеры при поставке: камера, шлейф присоединения, два диода ночного видения (рисунок 2.9).

Raspberry Pi и компьютерное зрение. Raspberry Pi был разработан как недорогой одноплатный компьютер с целью продвижения обучения информатике в школах. Он также представляет собой долгожданное возвращение к простому и увлекательному, но эффективному способу изучения информатики и программирования.

Рисунок 2.9 – Камера высокого разрешения

Вы можете использовать Raspberry Pi для изучения и реализации концепций в компьютерном зрении. С компьютером за $ 35 Raspberry Pi и веб-камерой USB каждый может позволить себе быстро стать профессионалом в области компьютерного зрения и создать реальное приложение для компьютерного зрения, чтобы произвести впечатление на друзей и коллег.

Цветовые пространства и преобразования. Цветовое пространство - это математическая модель, используемая для представления цветов. Обычно цветовые пространства используются для представления цветов в числовой форме и для выполнения математических и логических операций с ними. В этой книге мы в основном используем следующие цветовые пространства: BGR (стандартное цветовое пространство OpenCV), RGB, HSV и оттенки серого. BGR означает синий, зеленый и красный. HSV представляет цвета в форматах Hue, Saturation и Value. OpenCV имеет функцию cv2.cvtColor (img, conv_flag), которая позволяет нам изменять цветовое пространство изображения (img), в то время как исходное и целевое цветовые пространства указываются в параметре conv_flag.

Особенности камеры fisheye:

1) Широкоугольный объектив рыбий глаз, позволяющий охватить большую площадь поверхности;

2) Камера ночного видения для 130/160 градусов, возможность распознавания в различных условиях освещения;

3) 5 Мп пикселей

4) Поддержка разрешения 1080P, благодаря чему система сможет обрабатывать более четкое изображение;

5) Легкая в установке.

Технические характеристики:

Raspberry Pi Камера

5-мегапиксельная OV5647 датчик

Камера Технические характеристики

CCD Размер: 1/4 дюйма

Лучшее разрешение датчика: 1080 p

4 Винтовых отверстий

Используется для крепления

Фокусное растояние: 2,1

Диагональный угол: 130 градусов/160 градусов

Обеспечивает выходную мощность 3,3 В

Размер: 25 мм x 24 мм

## 2.5 Схема управления системы

Система распознавания, должна включать в себя два основных устройства: камера и микропроцессор.

Схема цифровой камеры будет представлена следующим образом (рисунок 2.10)

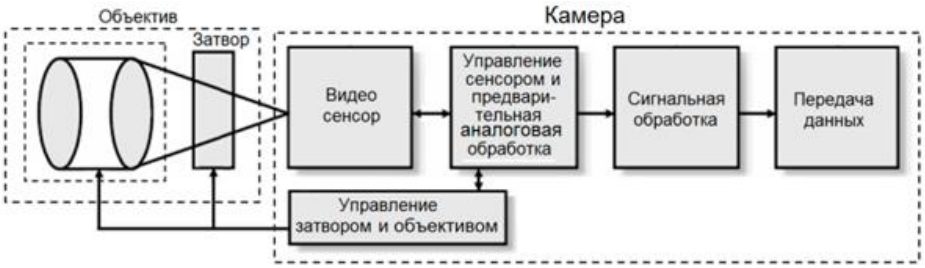


Рисунок 2.10 - Схема работы широкоугольной камеры

В то время, как схемы работы операционной системы, для выполнения стандартных вычислительных, а также задач загрузки и хранения представлена на рисунках ниже (рисунки 2.11-2.13).

Основные функции:

- исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).
- загрузка программ в оперативную память и их выполнение.
- стандартизованный доступ к периферийным устройствам (устройства ввода-вывода).
- управление оперативной памятью (распределение между процессами, организация виртуальной памяти).
- управление доступом к данным на энергонезависимых носителях (таких как жёсткий диск, оптические диски и др.), организованным в той или иной файловой системе.
- обеспечение пользовательского интерфейса.
- сохранение информации об ошибках системы.

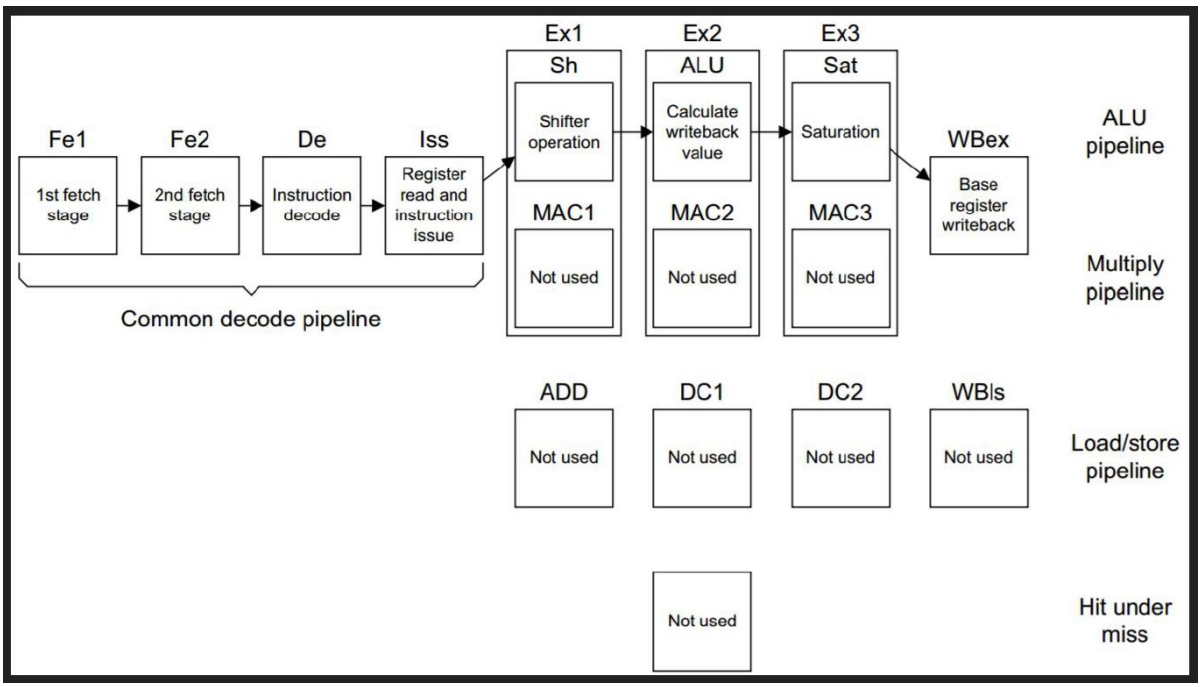

Рисунок 2.11 – Схема работы для вычисления стандартных алгебраических задач

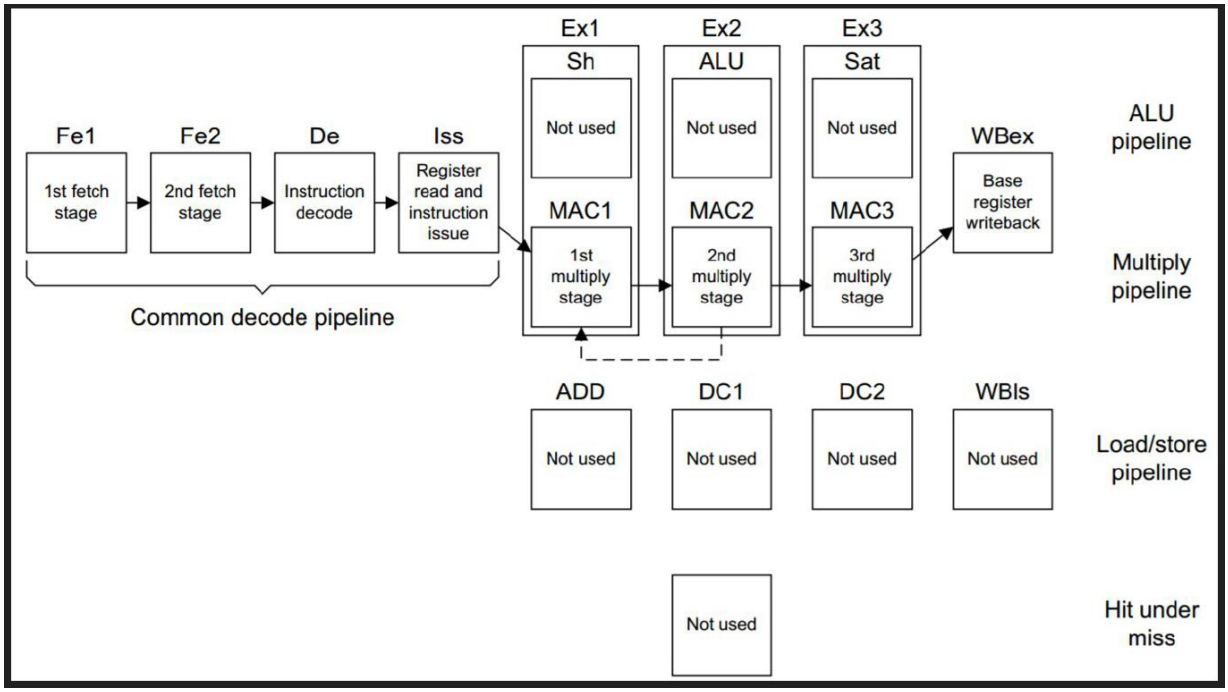


Рисунок 2.12 – Схема работы для вычисления стандартных задач умножения

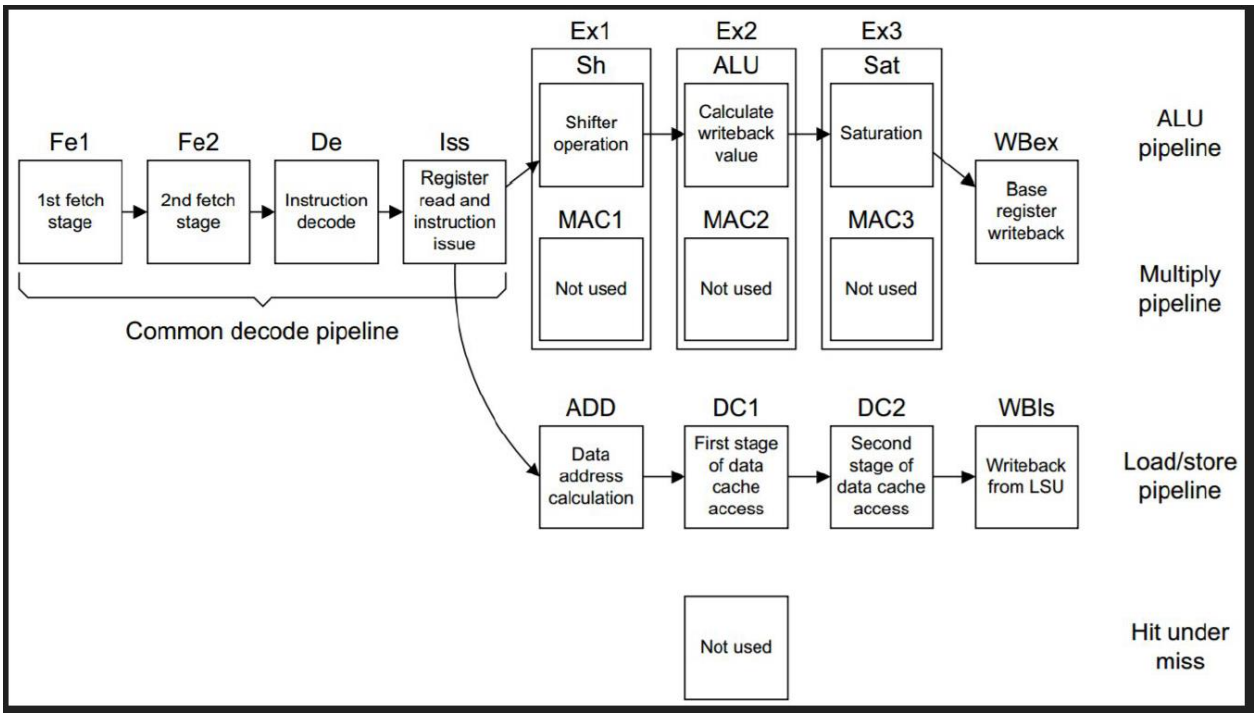


Рисунок 2.13 – Схема работы для вычисления стандартных задач загрузки и хранения данных

Так как система хранения данных кэша в системе представлена одноуровневой системой памяти, мы изобразим её на следующей схеме (рисунок 2.14).

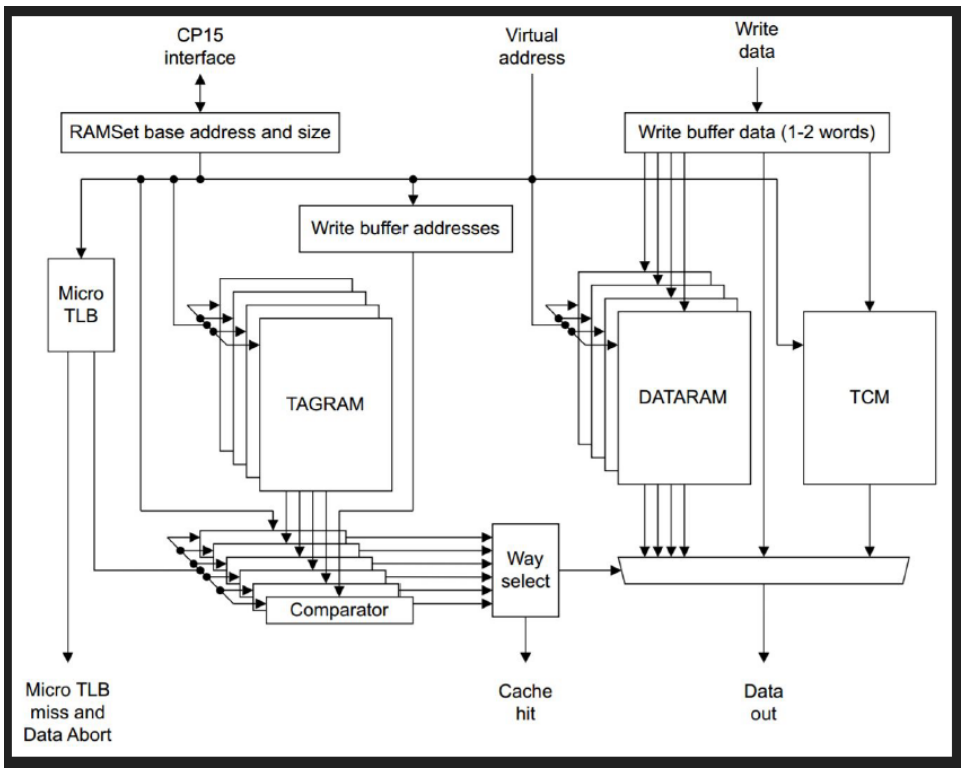


Рисунок 2.14 – Структура организации кэша

## 3 Программное обеспечение

### 3.1 Среда разработки и листинг нейросети

Лучший способ понять разработку нейросетей, применяемый к компьютерным задачам, — это просто визуализировать данный ввод, который будет расширен и искажен. Чтобы выполнить эту визуализацию, давайте создадим простой скрипт на Python, который использует встроенную мощь Keras для выполнения дополнения данных. Создайте новый файл, назовите его augmentation_demo.py. и вставьте следующий код (рисунок 3.1):

```
1 # import the necessary packages
2 from keras.preprocessing.image import ImageDataGenerator
3 from keras.preprocessing.image import img_to_array
4 from keras.preprocessing.image import load_img
5 import numpy as np
6 import argparse
```

Рисунок 3.1 – Импортирование библиотек.

Обратите внимание на строку 2, куда мы импортируем класс ImageDataGenerator из Keras - этот код будет использоваться для дополнения данных и включает все соответствующие методы, помогающие нам преобразовать наше входное изображение.

Далее, мы анализируем наши аргументы командной строки (рисунок 3.2):

```
8 # construct the argument parse and parse the arguments
9 ap = argparse.ArgumentParser()
10 ap.add_argument("-i", "--image", required=True,
11 help="path to the input image")
12 ap.add_argument("-o", "--output", required=True,
13 help="path to output directory to store augmentation examples")
14 ap.add_argument("-p", "--prefix", type=str, default="image",
15 help="output filename prefix")
16 args = vars(ap.parse_args())
```

Рисунок 3.2 – Анализ аргументов

Наш скрипт требует три аргумента командной строки, каждый из которых подробно описан ниже:

- image: это путь к входному изображению, к которому мы хотим применить увеличение данных и визуализировать результаты.

- output: после применения дополнения данных к данному изображению мы хотели бы сохранить результат на диске, чтобы мы могли проверить его - этот переключатель управляет выходным каталогом.

- prefix: строка, которая будет добавлена к имени файла выходного изображения.

Теперь, когда наши аргументы командной строки проанализированы, давайте загрузим наше входное изображение, преобразуем его в Keras-совместимый массив и добавим к изображению дополнительное измерение, как мы делали бы, если бы мы готовили наше изображение для классификации:

```
18 # load the input image, convert it to a NumPy array, and then
19 # reshape it to have an extra dimension
20 print("[INFO] loading example image...")
21 image = load_img(args["image"])
22 image = img_to_array(image)
23 image = np.expand_dims(image, axis=0)
```

Рисунок 3.3 – Загрузка входного изображения

Теперь мы готовы инициализировать наш ImageDataGenerator:

```
25 # construct the image generator for data augmentation then
26 # initialize the total number of images generated thus far
27 aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
28 height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
29 horizontal_flip=True, fill_mode="nearest")
30 total = 0
```

Рисунок 3.4 – Анализ изображения ImageDataGenerator

Вместо этого мы сосредоточимся на параметрах дополнения, которые мы, будем использовать в своем собственном приложении. Параметр вращение_диапазона контролирует диапазон степеней случайных вращений. Здесь мы допустим случайное вращение нашего входного изображения на 30 градусов. И width_shift_range, и height_shift_range используются для горизонтального и вертикального сдвигов соответственно.

Значение параметра является долей данного измерения, в данном случае 10%. Диапазон сдвига управляет углом в направлении против часовой стрелки в радианах, в которых будет разрешено срезать наше изображение. Затем у нас есть zoom_range, значение с плавающей запятой, которое позволяет «увеличивать» или «уменьшать» изображение в соответствии со следующим равномерным распределением значений: [1 - zoom_range, 1 + zoom_range].

И, наконец, логическое свойство horizontal_flip контролирует, разрешено ли переключение данного входа по горизонтали в процессе обучения. Для большинства приложений компьютерного зрения горизонтальное отражение изображения не меняет результирующую метку класса, но есть приложения, в которых горизонтальное (или вертикальное) изменение меняет смысловой смысл изображения. Будьте осторожны при применении этого типа дополнения данных, так как наша цель состоит в том, чтобы немного изменить входное изображение, создавая тем самым новый обучающий образец, не изменяя саму метку класса, как и было исследовано в разделе 1.4.

Сделав несколько пробных заходов, мы можем скомпилировать и записать листинг, нашей нейронной сети, использующей сверточную архитектуру, и библиотеки Torch7 и OpenCV (рисунки 3.5-3.6).

```
--
useOpenCV=true
require 'nn'
require 'image'
require 'camera' -- for camera
display = require 'display' -- for displaying


local display_sample_in={}
local display_output={}

torch.setdefaulttensortype('torch.FloatTensor')

-- load pre-trained model
local prefix=os.getenv("HOME")

local t = torch.Timer()
local m=torch.load(prefix..'nin_bn_final_arm.t7')
print(string.format("loading model:%.2fsec",t:time().real))

local model=m:unpack()

-- add soft max layer, to ouput pseudo-probabilities
model:add(nn.LogSoftMax())
-- switch model to evaluate mode
model:evaluate()

local classes=model.classes
local words=torch.load(prefix..'words_1000_ascii.t7','ascii')

local mean_std=model.transform

-- input (from camera)
local iW=320
local iH=240

-- output (for model)
local oW=224
local oH=224

local topk=5

-- parameters for cropping:
local w1 = math.ceil((iW-oW)/2)
local h1 = math.ceil((iH-oH)/2)

local cam = image.Camera {idx=0,width=iW,height=iH}
```

Рисунок 3.5 – Внедрение интерфейса дисплея в интерфейс камеры

```
function
createModel(nGPU)
                    require 'cunn'

                    local model = nn.Sequential()

                    local function block(...)
                       local arg = {...}
                       local no = arg[2]
                       model:add(nn.SpatialConvolution(...))
                       model:add(nn.SpatialBatchNormalization(no,1e-3))
                       model:add(nn.ReLU(true))
                       model:add(nn.SpatialConvolution(no, no, 1, 1, 1, 1, 0, 0))
                       model:add(nn.SpatialBatchNormalization(no,1e-3))
                       model:add(nn.ReLU(true))
                       model:add(nn.SpatialConvolution(no, no, 1, 1, 1, 1, 0, 0))
                       model:add(nn.SpatialBatchNormalization(no,1e-3))
                       model:add(nn.ReLU(true))
                    end

                    local function mp(...)
                       model:add(nn.SpatialMaxPooling(...))
                    end

                    block(3, 96, 11, 11, 4, 4, 5, 5)
                    mp(3, 3, 2, 2, 1, 1)
                    block(96, 256, 5, 5, 1, 1, 2, 2)
                    mp(3, 3, 2, 2, 1, 1)
                    block(256, 384, 3, 3, 1, 1, 1, 1)
                    mp(3, 3, 2, 2, 1, 1)
                    block(384, 1024, 3, 3, 1, 1, 1, 1)

                    model:add(nn.SpatialAveragePooling(7, 7, 1, 1))
                    model:add(nn.View(-1):setNumInputDims(3))
                    model:add(nn.Linear(1024,1000))
                    model:add(nn.LogSoftMax())

                    model.imageSize = 256
                    model.imageCrop = 224

                    return model:cuda()
                 end
```

Рисунок 3.6 – Модель натренированной нейронной сети

### 3.2 Блок схема и программа для распознавания

На блок-схеме, программы обучения нейросети, указанной ниже можно четко обозначить все этапы её выполнения (рисунок 3.7).

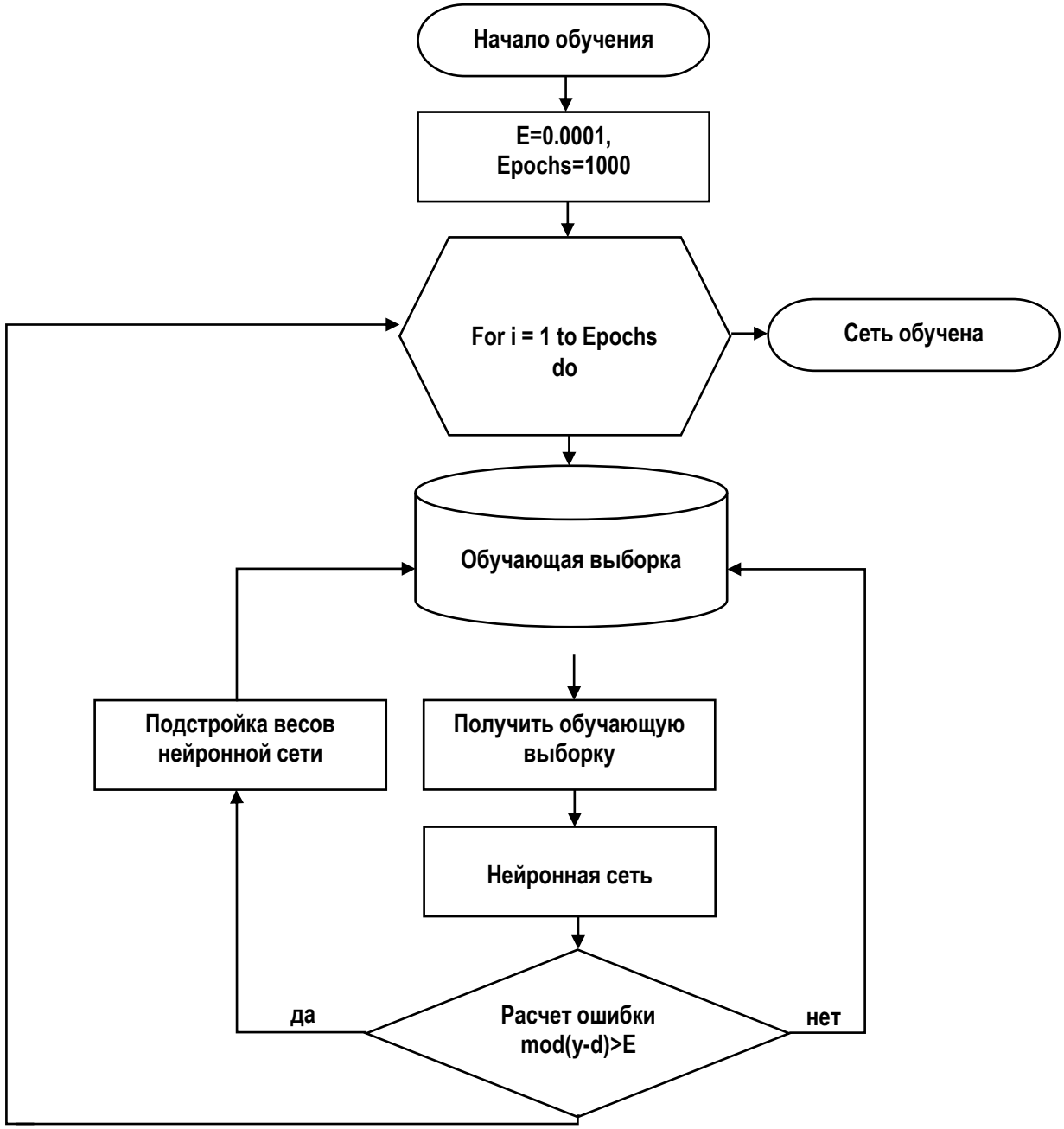


Рисунок 3.7 – Блок-схема алгоритма обучения

Основываясь на новой структуре нейронов и применения вероятностно-статистических анализов были получены высокие результаты распознавания. Данный граф показывает структурную схему построенной искусственной самоорганизующейся нейронной сети (ИСНС) для кластеризации данных.

Ниже приводится более детализированная блок-схема нейронной сети (рисунок 3.8).

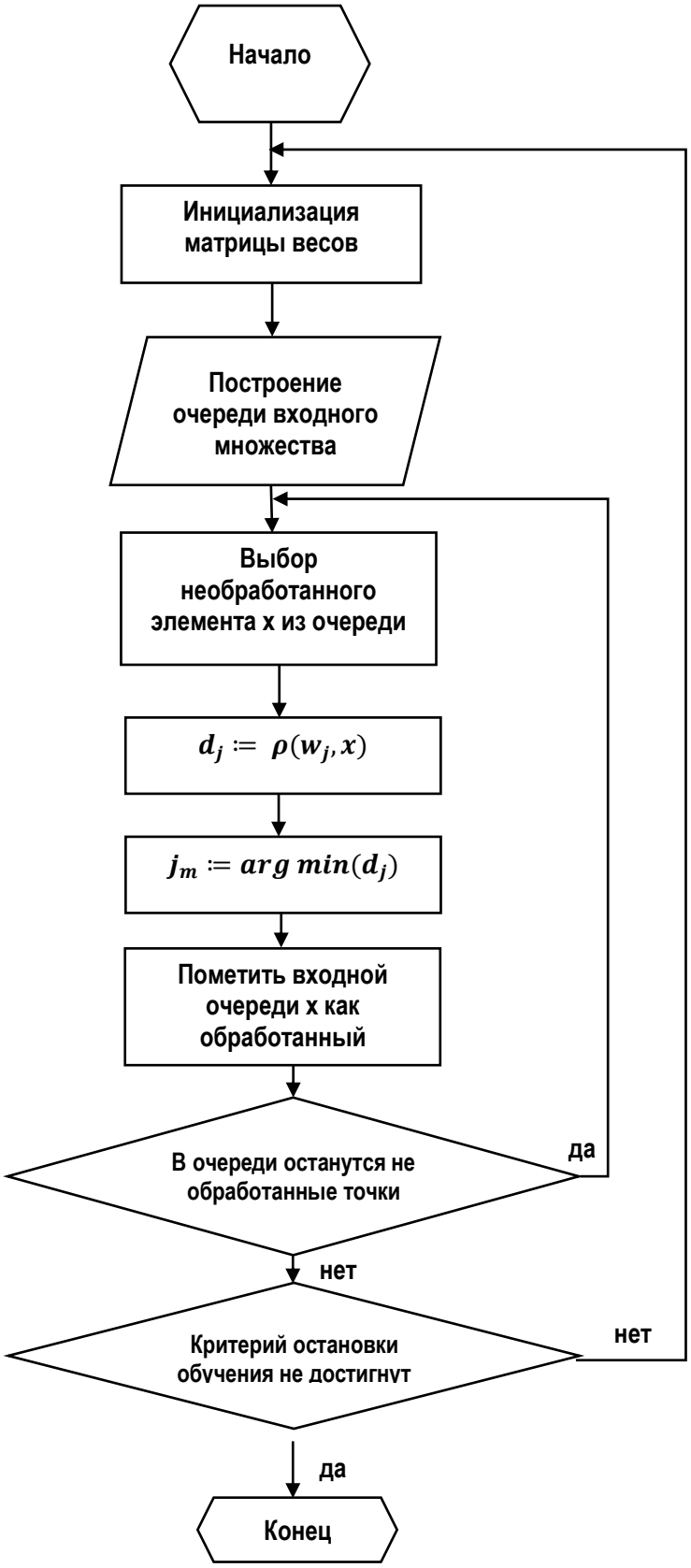


Рисунок 3.8 – Детализированная блок-схема распознавания нейронной сети.

В предыдущем разделе был указан листинг разработанной нейросети, и следующим шагом является проверка её работоспособности.

Проведем тест, изучим результат (рисунки 3.9 – 3.10)

Рисунок 3.9 – Тестовая установка

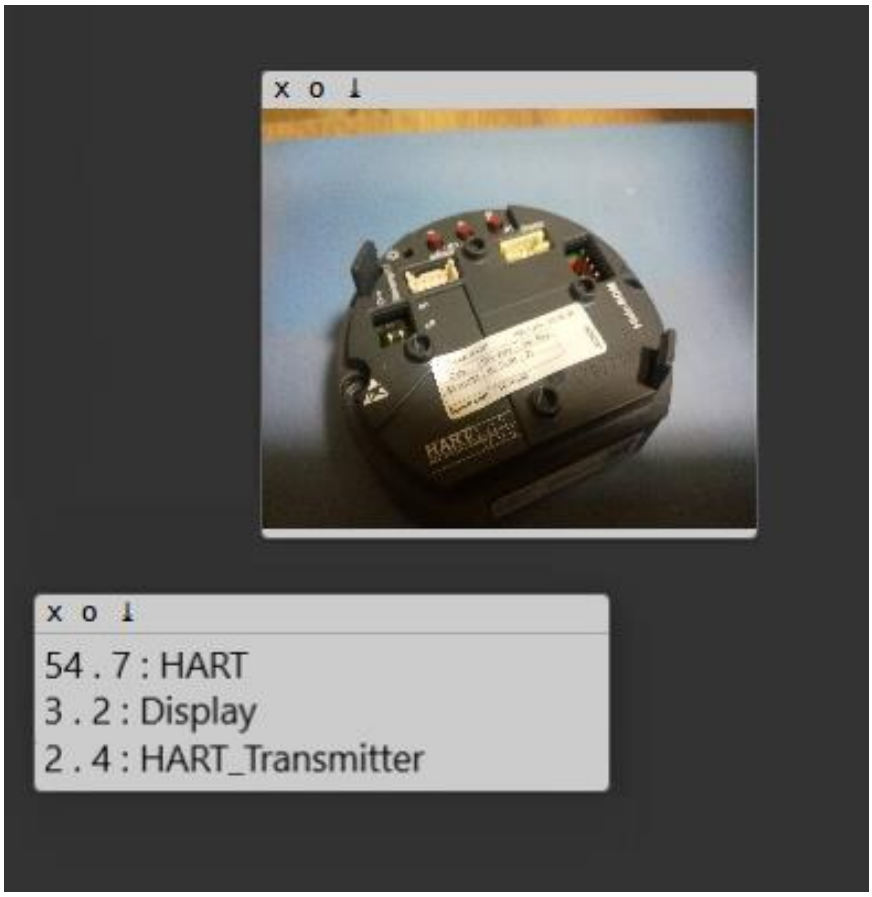


Рисунок 3.10 – Результат распознавания

## 4 Безопасность жизнедеятельности (БЖД)

### 4.1 Основание актуальности дипломного проекта

Универсальным средством, обеспечивающим комплексную автоматизацию производственных процессов и позволяющим быстро изменять последовательность, скорость и вид манипуляционных действий, являются промышленные роботы (ПР).

Механизация и автоматизация процессов, осуществляемая с помощью ПР, позволяет высвободить значительное число вспомогательных рабочих и направить их в основное производство. С помощью ПР успешно автоматизируются монотонно повторяющиеся операции и переходы производственного цикла, протекающие в производственной среде с высокой температурой, неприятными запахами, пылью, газами и гарью. Важным параметром, обеспечивающим безопасность персонала, обслуживающего ПР, и удобство работы оператора, является скорость перемещения исполнительных устройств. Многие промышленных производств, как правило, предполагают применение автоматизированного технологического процесса, которые, несомненно, должны соответствовать современным требованиям безопасности. Предприятия должны обустраивать рабочие места и выполнять требования безопасности, с целью максимизировать эффективной работы на предприятии без угрозы жизни сотрудников.

### 4.2 Обоснование дипломного проекта

Опираясь на все исследования, проведенные при создании данного дипломного проекта, можно с уверенностью заявить об его актуальности, которую можно подкрепить многими факторами. Во-первых: системы искусственного интеллекта уже давно доказали свою эффективность в управлении технологическими процессами; во-вторых: существует главное отличие применения именно машинного обучения от программирования контроллера на прямую, а именно самостоятельное предсказание атрибутов будущих объектов производства на основании уже изученных материалов, т.е. искусственный интеллект будет самостоятельно распознавать новые объекты, опираясь на свой опыт; в-третьих: в случае использования данной технологии, пропадает необходимость непосредственного нахождения оператора рядом с установкой, что в разы повышает эффективность системы.

Главным же новшеством данного проекта является универсальность в её применении, тот же искусственный интеллект, имея новую базу данных объектов, можно обучить распознавать совершенно различные объекты. К примеру, ИИ управляющее сортировкой деталей датчиков давления, можно переквалифицировать в сортировщика продуктов питания на совершенно другом производстве. Именно возможность применения одного обучающегося алгоритма в различных отраслях промышленности и делает систему, в некотором роде уникальной.

### 4.3 Улучшения в сфере безопасности

Главным рассматриваемым фактором, влияющим на безопасность жизнедеятельности в данном проекте, будет являться разработка и анедрение высокотехнологичной системы управления технологическим процессом сортировки деталей. Основными положительными факторами будут:

- при внедрении системы, отпадает необходимость нахождения оператора непосредственно рядом с узлом сортировки.
- система обуславливает наличие пульта управления, что значительно уменьшает риск получения травмы;
- отпадает необходимость в содержании большого штата сотрудников на линии сортировки, что уменьшит количество несчастных случаев на производстве;
- так как система запущена на микрокомпьютере, увеличивается мобильность установки, что дает возможность быстро переместить;
- сокращение социальных конфликтов ввиду малого количества штата и беспрерывной работы системы.

Но, при использовании высокотехнологичной системы управления технологическим процессом сортировки деталей появятся и затратные моменты:

- необходимость квалифицированного технического обслуживания обученным персоналом;

### 4.4 Обоснование темы дипломного проекта

В данном дипломном проекте была разработана интеллектуальная система технологического процесса сортировки на базе микрокомпьютера для ТОО «Process Automation». Результатом решения данной задачи является программное обеспечение, которое должно обеспечивать бесперебойную работу узла сортировки деталей.

Так как главной темой проекта является разработка программного обеспечения, в данном разделе будут рассмотрены вопросы:

1) определение оптимальных условий труда инженера - программиста;
2) расчет освещенности;
3) расчет уровня шума.

#### 4.4.1 Характеристика условий труда программиста

На сегодняшний день, прогресс в сфере науки и техники вносит все новые и новые трудности в условия работников умственного труда. Умственный труд стал требовать большей затраты сил, увеличения интенсивности, и затрачивает всё больше умственной, эмоциональной и физической энергии.

Это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

И с уверенностью можно сказать, что компьютерная техника затронула все области жизни человека. Но при этом нельзя завывать и о том, что при работе с компьютером человек подвергается воздействию ряда опасных и вредных производственных факторов: электромагнитных полей (диапазон радиочастот: ВЧ, УВЧ и СВЧ), инфракрасного и ионизирующего излучений, шума и вибрации, статического электричества и др.

Естественно, такая работа идет бок о бок с повышенной утомляемостью, психической, эмоциональной нагрузкой, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой ЭВМ. Именно поэтому огромное значение имеет практичная и удобная в использовании конструкция, правильное расположение рабочих элементов, и конечно же, само рабочее место оператора.

Исходя из всех вышеперечисленных факторов, человеку, чья работа напрямую связана с компьютером, надо строго соблюдать регламентированный режим работы и отдыха. Иначе, у работников наблюдаются ухудшение зрения из-за постоянного напряжения зрительного аппарата, снижение эффективности, головные боли, повышенная раздражительность, проблемы с режимом сна, чрезмерная утомляемость, и боли в глазах, области шеи, руках и пояснице.

4.4.2 Требования к производственным помещениям

Цвета окраски помещений и находящейся в них мебели, должны благоприятно влиять на зрительное восприятие и в целом создавать благополучные условия для сотрудников.

В зависимости от ориентации окон рекомендуется следующая окраска стен и пола:

окна ориентированы на юг: - стены зеленовато-голубого или светло-голубого цвета; пол - зеленый;

окна ориентированы на север: - стены светло-оранжевого или оранжево-желтого цвета; пол - красновато-оранжевый;

окна ориентированы на восток: - стены желто-зеленого цвета;

пол зеленый или красновато-оранжевый;

окна ориентированы на запад: - стены желто-зеленого или голубовато-зеленого цвета; пол зеленый или красновато-оранжевый.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения: для потолка: 60…70%, для стен: 40…50%, для пола: около 30%. Для других поверхностей и рабочей мебели: 30…40%.

Также нельзя забывать о лучшем для благоприятных условий труда - освещении. По-умному спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое

воздействие на работающего, повышает безопасность труда и снижает травматизм.

Различают несколько видов освещения:

а) е*стественное освещение* - освещение помещений дневным светом, это можно назвать практикой размещения окон, других проемов и отражающих поверхностей, чтобы солнечный свет (прямой или непрямой) может обеспечить эффективное внутреннее освещение. Особое внимание уделяется дневному освещению при проектировании здания, когда цель состоит в том, чтобы максимизировать визуальный комфорт или уменьшить потребление энергии. Экономия энергии может быть достигнута за счет сокращения использования искусственного (электрического) освещения или пассивного солнечного отопления. Использование энергии искусственного освещения можно уменьшить, просто установив меньшее количество электрических ламп в местах, где присутствует дневной свет, или автоматически уменьшая выключая электрические лампы в ответ на присутствие дневного света - процесс, известный как сбор дневного света.

б) искусственное освещение применяется при работе в темное время суток и днем, когда не удаётся обеспечить нормированные значения коэффициента естественного освещения (пасмурная погода, короткий световой день). Освещение, при котором недостаточное по нормам естественное освещение дополняется искусственным, называется совмещённым освещением. Это может сэкономить энергию вместо использования искусственного освещения, которое представляет собой основной компонент потребления энергии в зданиях. Правильное освещение может улучшить выполнение задач, улучшить внешний вид помещения или оказать положительное психологическое воздействие на сотрудников.

в) комбинированное - освещение, при котором к общему добавляется местное освещение.

Искусственное освещение подразделяется на рабочее, аварийное, эвакуационное, охранное. Рабочее освещение, в свою очередь, может быть общим или комбинированным. Общее - освещение, при котором светильники размещаются в верхней зоне помещения равномерно или применительно к расположению оборудования. Нормированный уровень освещения, в соответствии с санитарными нормами РК, приведен в таблице 4.1.

Таблица 4.1 - Уровень освещенности на рабочих местах

| Период года | Категория работ | Температура воздуха не более | Относительная влажность воздуха, % | Скорость движения воздуха, метров в секунду |
|---|---|---|---|---|
| Холодный | Легкая 1а | 22–24 | 40–60 | 0,1 |
| | Легкая 1б | 23–21 | 40–60 | 0,1 |
| Теплый | Легкая 1а | 23–25 | 40–60 | 0,1 |

| | | | | |
|---|---|---|---|---|
| Теплый | Легкая 1б | 22–24 | 40–60 | 0,1 |

Освещенность рабочих помещений с дисплеями рекомендуется в пределах 300–500 люкс. В поле зрения, работающего с дисплеем, не должны находиться окна и осветительные приборы. Светильники должны быть с рассеивателями, отражение на экране от источника света снимается установкой защитных экранов. Яркость свечения не должна быть меньше освещенности рабочей поверхности с документами, поскольку скачки яркости при смене полей зрения (с документа на экран и наоборот) должно быть минимальными. Оконные проемы в помещениях с персональными компьютерами должны быть оборудованы регулируемыми светозащитными устройствами (жалюзи, занавеси, внешние козырьки и др.).

Также важная роль отведена микроклимату. Согласно санитарным нормам микроклимата, в кабинах, пультах и постах управления технологическими процессами, в залах вычислительной техники, а также в других помещениях при выполнении работ операторского типа, связанных с нервно-эмоциональным напряжением, должны соблюдаться оптимальные величины температуры воздуха (22–24°С), его относительной влажности (60–40%) и скорости движения (не более 0,1 м/с) (табл. 1). Перечень других производственных помещений, в которых должны соблюдаться оптимальные нормы микроклимата, определяется отраслевой документацией, согласованной с органами санитарного надзора в установленном порядке.

Нормы температуры и влажности в помещении устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (таблица 2).

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше 19,5м3/человека с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в таблице 4.2.

Таблица 4.2 (а) - Параметры микроклимата для помещений, где установлены компьютеры

| Период | Параметр микроклимата | Величина |
|---|---|---|
| Холодный | Температура воздуха в помещении<br>Относительная влажность<br>Скорость движения воздуха | 22…24°С<br>40…60%<br>до 0,1м/с |
| Теплый | Температура воздуха в помещении<br>Относительная влажность<br>Скорость движения воздуха | 23…25°С<br>40…60%<br>0,1…0,2м/с |

И не стоит забывать о воздействии шума и вибрации, которые также имеют значительное влияние на условия работы. Чрезмерный шум может привести к ухудшению слуха. Шум, который содержит внезапные, мощные

ударные звуки, так называемый импульсный шум, особенно вреден. Он является наиболее распространенной причиной профессиональных заболеваний. Люди могут находить даже низкий уровень шума беспокоящим, например, при выполнении работы, которая требует концентрации в офисах открытой планировки. Вредные вибрации — это вибрации тела, передаваемые от инструмента к руке, или вибрации всего тела, передаваемые человеку с платформы, например, сиденья промышленной машины.

Таблица 4.2 (б) - Нормы подачи свежего воздуха в помещения, где расположены компьютеры

| Характеристика помещения | Объемный расход подаваемого в помещение свежего воздуха, $м^3$ /на одного человека в час |
|---|---|
| Объем до 20 $м^3$ на человека<br>20…40 $м^3$ на человека<br>Более 40 $м^3$ на человека | Не менее 30<br>Не менее 20<br>Естественная вентиляция |

В таблице 4.3 указаны предельные уровни звука в зависимости от категории тяжести и напряженности труда, являющиеся безопасными в отношении сохранения здоровья и работоспособности.

Таблица 4.3 - Предельные уровни звука, дБ, на рабочих местах.

| Категория напряженности труда | Категория тяжести труда | | | |
|---|---|---|---|---|
| | I. Легкая | II. Средняя | III. Тяжелая | IV. Очень тяжелая |
| I. Мало напряженный | 80 | 80 | 75 | 75 |
| II. Умеренно напряженный | 70 | 70 | 65 | 65 |
| III. Напряженный | 60 | 60 | - | - |
| IV. Очень напряженный | 50 | 50 | - | - |

Уровень шума на рабочем месте математиков-программистов и операторов видеоматериалов не должен превышать 50дБА, а в залах обработки информации на вычислительных машинах - 65дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами. Уровень вибрации в помещениях вычислительных центров может быть снижен путем установки оборудования на специальные виброизоляторы.

### 4.5 Требования к рабочему месту

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

Эргономическими аспектами проектирования видеотерминальных рабочих мест, в частности, являются: высота рабочей поверхности, размеры пространства для ног, требования к расположению документов на рабочем месте (наличие и размеры подставки для документов, возможность различного размещения документов, расстояние от глаз пользователя до экрана, документа, клавиатуры и т.д.), характеристики рабочего кресла, требования к поверхности рабочего стола, регулируемость элементов рабочего места.

Главные элементы рабочего места:

Моторное поле - пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук — это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом (рисунок 4.1).

Где 1 – оптимальная зона моторного поля;

2 – зона легкой досягаемости моторного поля;

3 – зона максимальной досягаемости моторного поля.

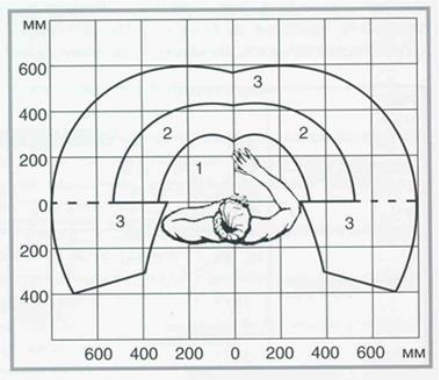


Рисунок 4.1 - Зоны досягаемости рук в горизонтальной плоскости

Оптимальное размещение предметов труда и документации в зонах досягаемости:

Дисплей размещается в зоне 3 (в центре);

Системный блок размещается в предусмотренной нише стола;

Клавиатура - в зоне 1;

Компьютерная - в зоне 1 справа;

Сканер в зоне 2 (слева);

Принтер находится в зоне 3 (справа);

Рабочая документация: необходимая при работе - в зоне легкой досягаемости ладони – 1, а в выдвижных ящиках стола - литература, неиспользуемая постоянно (рисунок 4.2).

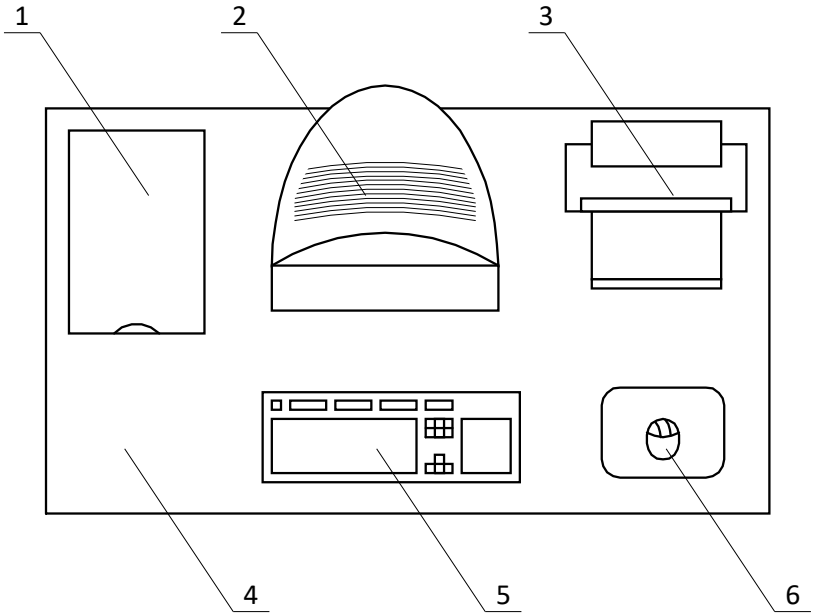


Рисунок 4.2 - Размещение основных и периферийных составляющих ПК (1 – сканер, 2 – монитор, 3 – принтер, 4 – поверхность рабочего стола, 5 – клавиатура, 6 – манипулятор типа «мышь»).

Для комфортной работы стол должен удовлетворять следующим условиям:

- высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;
- нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;
- поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;
- конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей).
- высота рабочей поверхности рекомендуется в пределах 680-760мм. Высота поверхности, на которую устанавливается клавиатура, должна быть около 650мм.

### 4.6 Режим труда

Как уже было неоднократно отмечено, при работе с персональным компьютером очень важную роль играет соблюдение правильного режима труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

В табл. 4.4 представлены сведения о регламентированных перерывах, которые необходимо делать при работе на компьютере, в зависимости от продолжительности рабочей смены, видов и категорий трудовой деятельности с ВДТ (видеодисплейный терминал) и ПЭВМ ( «Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работ»).

Таблица 4.4 - Время регламентированных перерывов при работе на компьютере

| Категория работы с ВДТ или ПЭВМ | Уровень нагрузки за рабочую смену при видах работы с ВДТ | | | Суммарное время регламентированных | |
|---|---|---|---|---|---|
| | Группа А, количество знаков | Группа Б, количество знаков | Группа В, часов | При 8-часовой смене | При 12-часовой смене |
| I | до 20000 | до 15000 | до 2,0 | 30 | 70 |
| II | до 40000 | до 30000 | до 4,0 | 50 | 90 |
| III | до 60000 | до 40000 | до 6,0 | 70 | 120 |

Примечание. Время перерывов дано при соблюдении указанных Санитарных правил и норм. При несоответствии фактических условий труда требованиям Санитарных правил и норм время регламентированных перерывов следует увеличить на 30%.

Эффективность перерывов повышается при сочетании с производственной гимнастикой или организации специального помещения для отдыха персонала с удобной мягкой мебелью, аквариумом, зеленой зоной и т.п.

### 4.7 Расчет освещенности

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света трех видов: ламп накаливания, люминесцентных ламп и светодиодных ламп. Будем использовать светодиодные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественному свету;
- более высокая световая отдача (до 150 Лм/Вт);
- в 5-10 раз меньшее энергопотребление;
- длительный срок службы (30-100 тысяч часов);
- устойчивость к ударам и вибрации;
- независимость световой отдачи и срока службы от напряжения;

Расчет освещения производится для помещения планировки типа прямоугольник площадью 30 м$^2$, ширина которой 5 м, высота - 3 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}$$

(4.1)

где $F$ - рассчитываемый световой поток, Лм;

$E$ - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет E = 300Лк;

$S$ - площадь освещаемого помещения (в нашем случае S = 30 м$^2$);

$Z$ - отношение средней освещенности к минимальной (обычно принимается равным 1,1…1,2 , пусть Z = 1,15);

$K$ - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K$ = 1,5);

$n$ - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемых коэффициентами отражения от стен ($P_C$) и потолка ($P_П$)), значение коэффициентов $P_C$ и $P_П$ были указаны выше: $P_C$=50%, $P_П$=70%. Значение n определим по таблице 5 коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = \frac{S}{h \cdot (A+B)} \tag{4.2}$$

, где $S$ - площадь помещения, S = 30 м$^2$;
$h$ - расчетная высота подвеса, h = 2.92 м;
$A$ - ширина помещения, A = 5 м;
$B$ - длина помещения, B = 6 м.
Подставив значения получим:

$$I = \frac{30}{2{,}92(5+6)} = 0{,}93$$

Таблица 4.4 - вспомогательная таблица для расчета освещенности

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | потолок | 0,8 | 0,7 | 0,7 | 0,5 | 0,5 | 0,5 |
| | стены | 0,5 | 0,5 | 0,3 | 0,5 | 0,3 | 0,3 |
| | пол | 0,3 | 0,3 | 0,3 | 0,3 | 0,3 | 0,1 |
| Индекс помещения | 0,60 | 0,33 | 0,32 | 0,25 | 0,3 | 0,24 | 0,24 |
| | 0,80 | 0,41 | 0,39 | 0,32 | 0,36 | 0,3 | 0,29 |
| | 1,00 | 0,47 | 0,45 | 0,38 | 0,42 | 0,35 | 0,34 |
| | 1,25 | 0,53 | 0,51 | 0,44 | 0,47 | 0,41 | 0,3 |
| | 1,50 | 0,58 | 0,55 | 0,48 | 0,51 | 0,45 | 0,43 |
| | 2,00 | 0,65 | 0,62 | 0,56 | 0,57 | 0,52 | 0,49 |
| | 2,50 | 0,7 | 0,67 | 0,61 | 0,61 | 0,56 | 0,53 |
| | 3,00 | 0,64 | 0,71 | 0,65 | 0,64 | 0,6 | 0,56 |
| | 4,00 | 0,79 | 0,75 | 0,7 | 0,68 | 0,64 | 0,6 |
| | 5,00 | 0,83 | 0,78 | 0,74 | 0,71 | 0,68 | 0,62 |

Примерно выберем коэффициент отражения поверхностей 0,7-0,5-0,3 (четвёртый столбик таблицы) соответствует помещению с белым потолком, светлыми стенами, и напольным покрытием, которое темнее стен (это наиболее распространённый вариант)

Зная индекс помещения $I$, по таблице находим $n$ = 0,45

Подставим все значения в формулу для определения светового потока $F$:

$$F = \frac{300*1{,}5*30*1{,}15}{0{,}45} = 34\ 500 \text{ Лм}$$

Для освещения выберем светодиодные, подвесные панели ПСВ-48W, световой поток которых F = 4000лм.

Рассчитаем необходимое количество ламп по формуле:

$$N = \frac{F}{F_л}$$

(4.3)

*N* - определяемое число ламп;

*F* - световой поток, F = 33750 Лм;

$F_л$- световой поток лампы, $F_л$ = 4000 Лм.

$$N = \frac{33750}{4000} = 8\ шт$$

Далее, начертим схему расположения светодиодных панелей в помещении (рисунок 4.3)

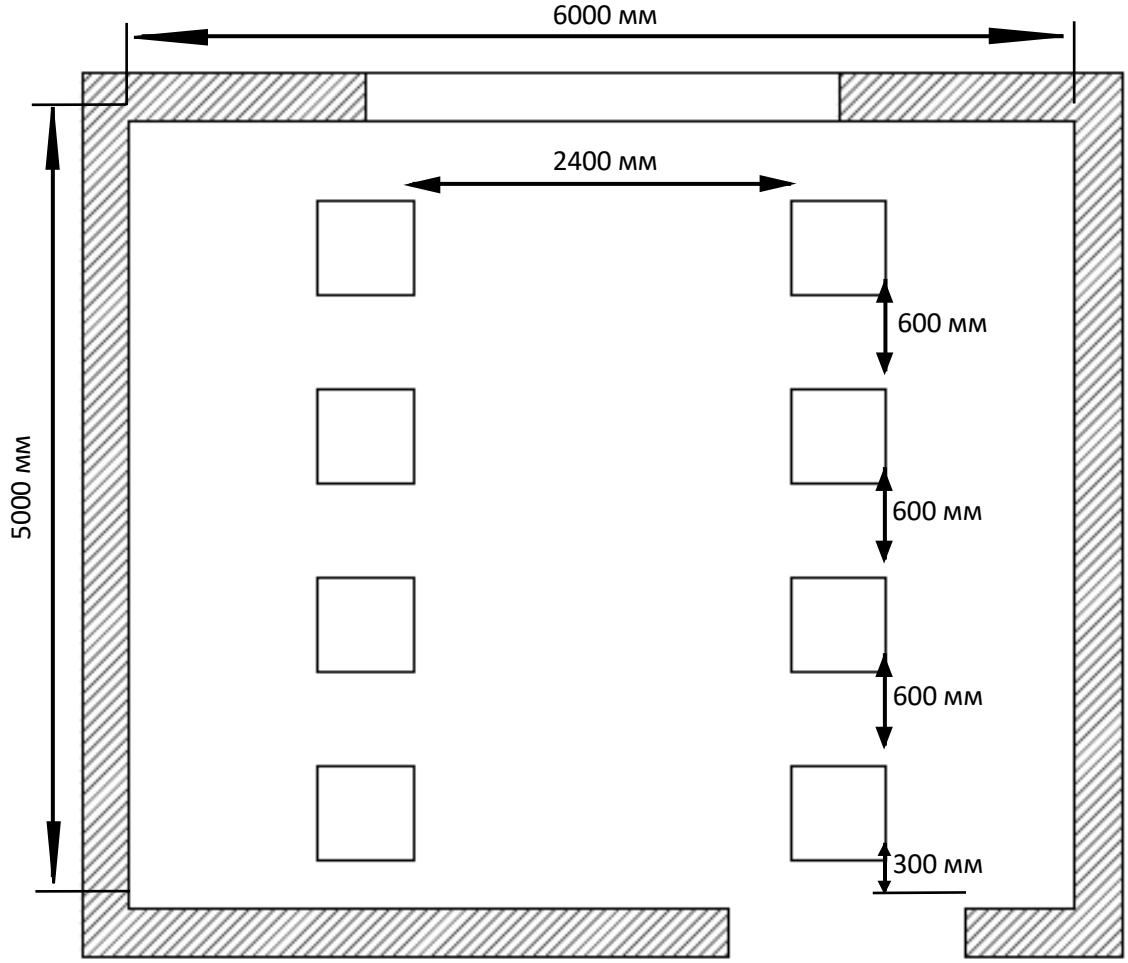


Рисунок 4.3 – Расположение светодиодных панелей в помещении

### 4.8 Расчет уровня шума

Одним из неблагоприятных факторов производственной среды в ИВЦ является высокий уровень шума, создаваемый печатными устройствами, оборудованием для кондиционирования воздуха, вентиляторами систем охлаждения в самих ЭВМ.

Для решения вопросов о необходимости и целесообразности снижения шума необходимо знать уровни шума на рабочем месте оператора.

Уровень шума, возникающий от нескольких некогерентных источников, работающих одновременно, подсчитывается на основании принципа энергетического суммирования излучений отдельных источников:

$$L_{\Sigma} = 10 \lg \sum_{i=1}^{i=n} 10^{0,1 L_i}, \tag{4.4}$$

где $L_i$ – уровень звукового давления i-го источника шума;

n – количество источников шума.

Полученные результаты расчета сравнивается с допустимым значением уровня шума для данного рабочего места. Если результаты расчета выше допустимого значения уровня шума, то необходимы специальные меры по снижению шума. К ним относятся: облицовка стен и потолка зала звукопоглощающими материалами, снижение шума в источнике, правильная планировка оборудования и рациональная организация рабочего места оператора.

Уровни звукового давления источников шума, действующих на оператора на его рабочем месте представлены в таблице 4.5.

Таблица 4.5 - Уровни звукового давления различных источников.

| Источник шума | Уровень шума, дБ |
|---|---|
| Жесткий диск | 40 |
| Вентилятор | 45 |
| Монитор | 17 |
| Клавиатура | 10 |
| Принтер | 45 |
| Сканер | 42 |

Обычно рабочее место оператора оснащено следующим оборудованием: винчестер в системном блоке, вентилятор(ы) систем охлаждения ПК, монитор, клавиатура, принтер и сканер.

Подставив значения уровня звукового давления для каждого вида оборудования в формулу, получим:

$$L_{\Sigma}=10\cdot\lg(10^{4}+10^{4,5}+10^{1,7}+10^{1}+10^{4,5}+10^{4,2}) = 49,5 \text{ дБ}$$

Полученное значение не превышает допустимый уровень шума для рабочего места оператора, равный 65 дБ (ГОСТ 12.1.003-83). И если учесть, что вряд ли такие периферийные устройства как сканер и принтер будут использоваться одновременно, то эта цифра будет еще ниже. Кроме того, при работе принтера непосредственное присутствие оператора необязательно, т.к. принтер снабжен механизмом автоподачи листов.

## Вывод

В данном разделе дипломной работы были изучены вопросы улучшения безопасности труда в результате внедрения системы, условий труда: таких как температура, влажность, освещенность, уровни шума и вибрации. Также были изложены требования к рабочему месту программиста: условия должны обеспечивать комфорт, благоприятно влиять на эффективность работы, способствовать улучшению концентрации. Еще нельзя не сказать о том, что сотрудники предприятий должны соблюдать регламентированный режим работы и отдыха, иначе, это может прямым образом сказаться на состоянии здоровья работника. В соответствии с рекомендациями Минздрава РК, согласно которым каждый сотрудник, в целях заботы о собственном здоровье должен делать короткие перерывы на отдых, а также делать гимнастику и разминку во избежание нарушений состояния зрительного аппарата, болей в шее, пояснице и руках.

Следующим шагом, опираясь на все изученные данные, был проведен расчет освещенности помещения, в котором проводилась разработка системы и уровня шума, издаваемого различными офисными приборами. Далее исходя из проведенных расчетов, можно сделать вывод, что условия работы программиста – разработчика полностью соответствуют всем требуемым нормам, в соответствии с санитарными требованиями к офисным помещениям и обеспечивает рекомендуемые показатели безопасности жизнедеятельности.

## 5 Экономическая часть

### 5.1 Основание актуальности дипломного проекта

На сегодня существует огромное множество предприятий, сделавших упор на различные инструменты для обработки данных, для того чтобы перейти в новейшую цифровую эпоху. Особенно большой потенциал имеется у отрасли, исследующей внедрение математической методологии и, само собой разумеющееся, набирающий популярность искусственный интеллект (ИИ), который можно использовать для улучшения бизнес-процессов, что может обеспечить уменьшение количества издержек и способствовать получению дополнительной прибыли.

Многие промышленных производств, как правило, предполагают применение автоматизированного технологического процесса, которые, несомненно, должны быть подкреплены различными экономическими показателями, как: затраты на единицу выпускаемой продукции, производительность и общая эффективность оборудования, издержки из-за его непосредственного простоя, брак, поломка товара.

### 5.2 Обоснование дипломного проекта

Опираясь на все исследования, проведенные при создании данного дипломного проекта, можно с уверенностью заявить об его актуальности, которую можно подкрепить многими факторами. Во-первых: системы искусственного интеллекта уже давно доказали свою эффективность в управлении технологическими процессами; во-вторых: существует главное отличие применения именно машинного обучения от программирования контроллера на прямую, а именно самостоятельное предсказание атрибутов будущих объектов производства на основании уже изученных материалов, т.е. искусственный интеллект будет самостоятельно распознавать новые объекты, опираясь на свой опыт; в-третьих: в случае использования данной технологии, пропадает необходимость непосредственного нахождения оператора рядом с установкой, что в разы повышает эффективность системы.

Главным же новшеством данного проекта является универсальность в её применении, тот же искусственный интеллект, имея новую базу данных объектов, можно обучить распознавать совершенно различные объекты. К примеру, ИИ управляющее сортировкой деталей датчиков давления, можно переквалифицировать в сортировщика продуктов питания на совершенно другом производстве. Именно возможность применения одного обучающегося алгоритма в различных отраслях промышленности и делает систему, в некотором роде уникальной.

### 5.3 Экономические показатели

Главным рассматриваемым фактором, влияющим на экономику данного проекта, будет являться внедрение высокотехнологичной системы управления технологическим процессом сортировки деталей. Основными положительными факторами будут:

- внедрение высокотехнологичной системы приведет к более эффективному процессу сортировки, уменьшению затрачиваемого времени;
- отпадает необходимость в содержании большого штата сотрудников на линии сортировки;
- отсутствие необходимости в дорогостоящем оборудовании, т.к. система работает на микрокомпьютере, что также имеет положительные моменты – малые затраты на энергообеспечение и мобильность при размещении оборудования;
- сокращение социальных конфликтов ввиду малого количества штата и беспрерывной работы системы.

Но, при использовании высокотехнологичной системы управления технологическим процессом сортировки деталей появятся и затратные моменты:

- необходимость квалифицированного технического обслуживания обученным персоналом;

### 5.4 Обоснование темы дипломного проекта

В данном дипломном проекте была разработана интеллектуальная система технологического процесса сортировки на базе микрокомпьютера для ТОО «Process Automation». Результатом решения данной задачи является программное обеспечение, которое должно обеспечивать бесперебойную работу узла сортировки деталей.

Целью данного раздела является расчет:

1) себестоимости проекта;
2) экономической эффективности проекта.

Для подсчета себестоимости проекта и экономической эффективности данного программного продукта, нужно знать следующие составляющие (Таблица 5.1):

А) расчет затрат на энергоресурсы;
Б) амортизационные отчисления;
В) расчет фонда заработной платы;
Г) прочие расходы.

Таблица 5.1 – Исходные данные

| Показатели | Единицы измерения | Значение |
|---|---|---|
| Отчисления на обязательное медицинское страхование в РК | % | 1 |

*Продолжение таблицы 5.1*

| | | |
|---|---|---|
| Социальный налог в РК | % | 9,5 |
| Сбор за соцобеспечение | % | 3,5 |
| Заработная плата специалиста (инженера, программиста) | тг./мес. | 200 000 |
| Заработная плата специалиста (руководителя) | тг./мес. | 300 000 |
| Заработная плата специалиста (эксперта) | тг./мес. | 250 000 |
| Премия | % | 30 |
| Фонд рабочего времени в 2019 году | дней | 246 |
| при 40-часовой рабочей неделе | часов | 1968 |
| Стоимость микрокомпьютера | тг | 34000 |
| Стоимость комплектующих | тг | 9800 |
| Стоимость корпуса и шарниров крепления | тг | 3000 |
| Показатели | Единицы измерения | Значение |
| Срок полезного использования микромпьютера | лет | От 5 до 7 |
| Затраты на текущий и профилактический ремонт микрокомпьютера (от стоимости компьютера) в год | % | 6 |
| Затраты на материалы, необходимые для эксплуатации микрокомпьютера (от стоимости компьютера) в год | % | 5 |
| Потребляемая мощность компьютера | кВт | 0,16 |
| Тариф на электроэнергию | тг./ кВт-час. | 23,00 |

| | | |
|---|---|---|
| Коэффициент распределения накладных расходов | % | 40 |

Согласно данным, взятых с сайта поиска вакансий в Казахстане «hh.kz» было проведено сравнение средней зарплаты программиста с опытом работы 1 и 3 года в г. Алматы (с высшим образованием). Заработная плата руководителя, с аналогичным опытом работы варьировалась в том же диапазоне.

Далее, покупка микрокомпьютера, корпуса и всех комплектующих, согласно данным сайта «satu.kz» составляла бы примерно: 45 000 – 58 000 тг. В то же время, срок их полезного использования и технические характеристики можно посмотреть на сайте производителя.

Также для расчета энергопотребления был взят за основу дифференцированный тариф 2 уровня, равный 23 тг./ кВт-час, заданный поставщиком - АО АлматыЭнергоСбыт на май 2020 года.

Длительность цикла в днях по каждому виду работы укрупнено можно определить по формуле

$$t_n = \frac{T}{g_n . 8K} \tag{5.1}$$

где Т – трудоёмкость этапа, нормо-час;
– количество исполнителей по этапу;
8 – продолжительность рабочего дня, час;
К – коэффициент выполнения норм времени (К=1,1).

Таблица 5.2 – Трудоемкость разработки системы

| Этапы разработки | Трудоемкость, час | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Руководитель проекта | | Программист | | Эксперт | | Машинное время | |
| | Наим. работ | Время | Наим. работ | время | Наим. работ | время | Наим. работ | Время |
| Формирование требование к ПО | | | | | Э1 | 2 | | 1 |
| Техническое задание | Р1 | 5 | | | | | | 4 |
| Разработка концепции ПО | | | П1 | 8 | | | | 8 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Эскизный проект | | | П2 | 10 | | | | 10 |
| Технический проект | | | П3 | 125 | | | | 125 |
| Рабочая документация | | | П4 | 50 | | | | 50 |
| Тестирование | | | П5 | 5 | | | | 5 |

*Продолжение таблицы 5.2*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Корректировка документации | | | П6 | 15 | | | | 15 |
| Приемка рабочей документации | | | | | Э2 | 10 | | |
| Контроль за ходом выполнения проекта | Р2 | 10 | | | | | | 5 |
| Ввод в действие | | | | | Э3 | 5 | | |
| Итого | | 15 | | 213 | | 17 | | 273 |

Далее, на основе данных трудоемкости процесса, можно построить график проведения всех работ (Рисунок 5.1). И только после построения сетевого графика, возможно сделать вывод, что по причине того, что рабочие не могут выполнять работы другого – нельзя оптимизировать график.

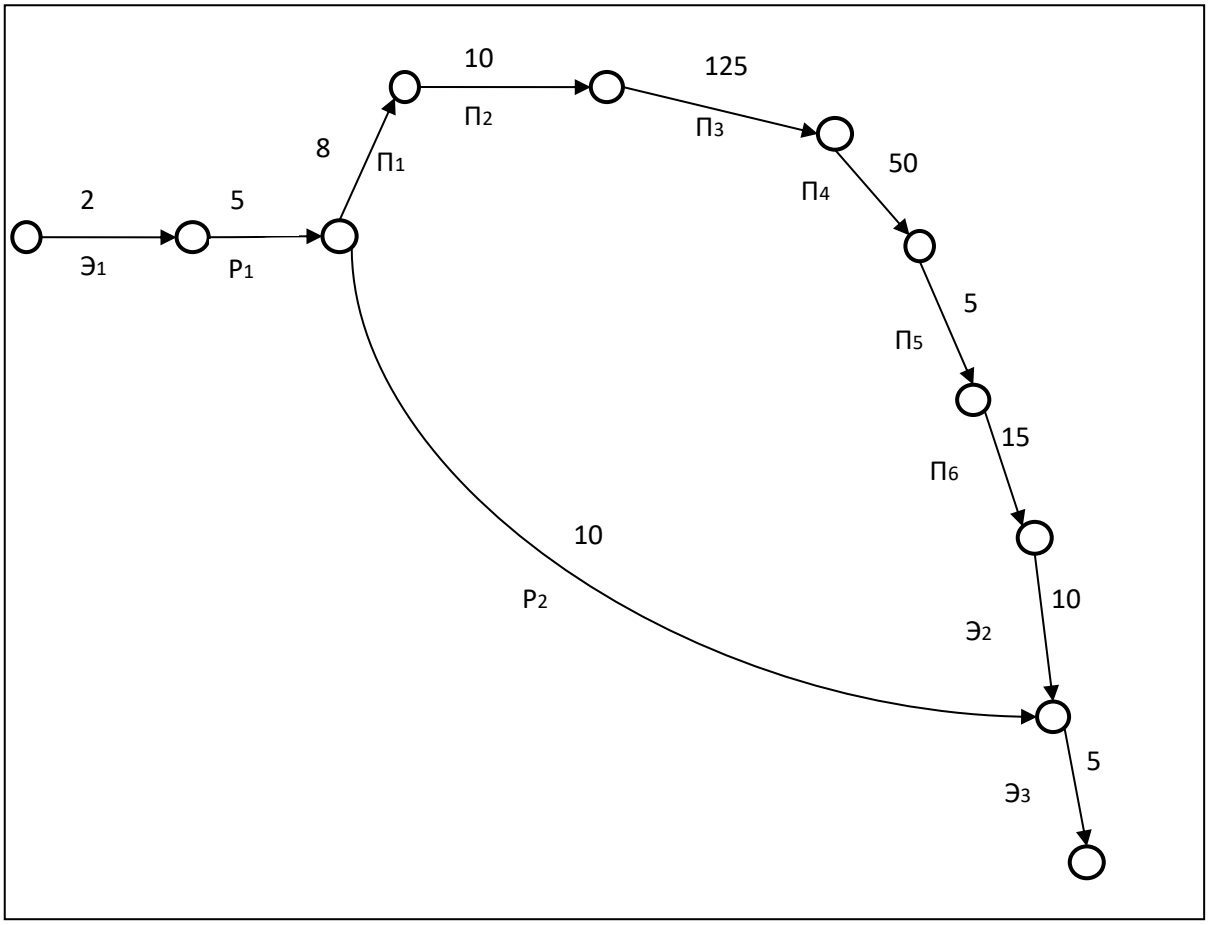


Рисунок 5.1 – Сетевой график

**5.5 Расчет затрат на разработку продукта**

1. Вычисление затрат на оплату труда (в случае привлечения разных специалистов определяется отдельно по каждому).

$$ЗП = ЗПср.час. * Тпр. \quad (5.2)$$

$ЗП_{(программист)}$ = 1585,36 * 213 = 337 681,68 тг

$ЗП_{(руководитель)}$ = 2378,04 * 15 = 35 670,6 тг

$ЗП_{(эксперт)}$ = 1981,7 *17= 33 688,9 тг

ЗП итог =337 681,68 + 35 670,6 + 33 688,9 = 407 041,18 тг

где Тпр – трудоемкость разработки программного продукта, час.;
ЗПср.час – среднечасовая оплата труда, тг/час.;

$$ЗПср.час. = О * Кп * 12 / Фрв, \quad (3)$$

ЗПср.час.(программист) = 200 000 * 1,3 * 12 / 1968 = 1585,36 тг/час

ЗПср.час.(руководитель) = 300 000 * 1,3 * 12 / 1968 = 2378,04 тг/час

ЗПср.час.(эксперт) = 250 000 * 1,3 * 12 / 1968= 1981,7 тг/час

где О – оклад специалиста по штатному расписанию,
Кп – процент премии;
Фрв – фонд рабочего времени в год.

2. Амортизационные отчисления

$$АО = (Сп+Ск+Скш) *На * Тмаш / Фрв, \quad (5.4)$$

АО = 46 800 * 0,167 * 273/ 1968= 1084, 17 тг

где Сп – стоимость микрокомпьютера, тг;
Ск – стоимость комплектующих;
Скш – стоимость креплений и шарниров;
На – норма амортизационных отчислений за год, %

$$На = 1 / СПИ * 100\% \quad (5.5)$$
$$На = 1 / 6 * 100\% = 16,7$$

СПИ – срок полезного использования микрокомпьютера, лет
Тмаш – машинное время, час
Фрв – фонд рабочего времени в год.

3. Затраты на текущий и профилактический ремонт компьютера

$$Зпр.р. = Сп * 0,06 * Тмаш / Фрв \quad (5.6)$$

Зпр.р. = 46 800 * 0,06 * 273 / 1968 = 389,52 тг

4. Затраты на материалы, необходимые для эксплуатации компьютера

$$Зматер. = Сп * 0,02 * Тмаш / Фрв \quad (5.7)$$

Зматер = 46 800 * 0,02 * 273 / 1968 = 129,84 тг

5. Затраты на электроэнергию

$$Зэл. = W * Сэл. * Тмаш \quad (5.8)$$

W – потребляемая мощность компьютера, кВт

Сэл. – тариф на электроэнергию, 23,00 тг/кВт-час

Зэл = 0,16 * 23,00 * 273 = 1 004,64 тг

6. Расчет социального налога и соц. отчислений

Согласно законодательству РК, на 2020 социальный налог, для каждого сотрудника ТОО, работающих по общеустановленному режиму налогообложения, будет рассчитываться по формуле:

$$СН = (ЗП – ОПВ – ВОСМС) * 0,095 – СО \quad (5.9)$$

где СН – социальный налог (ставка 9,5 %)

ОПВ – обязательный пенсионный взнос (10% от ЗП)

ВОСМС – взносы на обязательное социальное медицинское страхование (1% от ЗП)

СО – социальные отчисления (ставка 3,5%)

ЗП – заработная плата

В свою очередь, социальные отчисления могут быть рассчитаны по формуле:

$$СО = (ЗП – ОПВ) * 0,035 \quad (5.10)$$

$СО_{программист}$ = (200 000 – 200 000*0,1) * 0,035 = 6 300 тг

$СО_{руководитель}$= (300 000 – 300 000*0,1) * 0,035 = 9 450 тг

$СО_{специалист}$= (250 000 – 250 000*0,1) * 0,035 = 7 875 тг

Из этого следует:

$СН_{программист}$ = (200 000- 200 000*0,1-200 000*0,01)*0,095-6 300 = 10610 тг

$СН_{руководитель}$ = (300 000- 300 000*0,1-300 000*0,01)*0,095-9 450 =15915 тг

$СН_{специалист}$ = (250 000- 250 000*0,1-250 000*0,01)*0,095-7 875=11687,5 тг

$СО_{общ}$ = 6 300 + 9 450 + 7875 = 23 635 тг

$СН_{общ}$ = 10 610 + 15 915 + 11 687, 5 = 38 213 тг

Вып = $СО_{общ}$ + $СН_{общ}$ = 23 635 + 38 213 = 61 848тг

7. Расчет предполагаемой прибыли

Предположим, что плановая прибыль будет равна 25% от себестоимости проекта.

$$ПП = Сп*0,25 \quad (5.11)$$

ПП = 627 675,11 * 0,25 = 156 918,77 тг

7. Расчет цены интеллектуального труда
Цена – это себестоимость плюс чистый доход, т.е.

$$Ц = С + ПП \quad (5.12)$$

Ц = 627 675,11 + 156 918, 77 = 784 593,88 тг

где С – себестоимость продукта;
ПП – чистый доход.

$$Цп = С *(1 + Р / 100), \quad (5.13)$$

Цп = 628 126,7 * (1+ 40 / 100) = 878 745,154 тг,

где Р – рентабельность (20% - 40%);
Цп – первоначальная цена
Далее определяется цена реализации с учётом НДС

$$Цр = Цп + НДС. \quad (5.14)$$

Налог на добавленную стоимость (НДС) 12% , следовательно,

$$Цр = Цп * 1,12. \quad (5.15)$$

Цр = 879 377,38 * 1,12 = 984 194,57 тг

Таблица 5.3 – Расчет себестоимости и цены программного продукта

| Показатели | Условное обозначение | Значение |
|---|---|---|

*Продолжение таблицы 5.3*

| | | |
|---|---|---|
| 1. Прямые расходы, в том числе | | |
| 1.1 Затраты на оплату труда | ЗП | 407 041,18 |
| 1.2. Амортизационные отчисления | АО | 1084, 17 |
| 1.3. Затраты на текущий и профилактический ремонт | Зпр.р | 389,52 |
| 1.4. Затраты на материалы, необходимые для эксплуатации | Зматер | 129,84 |
| 1.5. Затраты на электроэнергию | Зэл | 1 004,64 |

| | | |
|---|---|---|
| 1.6 Затраты на соц. налог и соц. отчисления | Вып | 61 848 |
| 2. Косвенные (накладные) расходы | Знакл | 156 177,8 |
| 3. Итого затраты на разработку программного продукта (полная себестоимость) | | 627 675,11 |
| 4. Плановая прибыль (10-50% от полной себестоимости) | | 156 918,77 |
| 5. Цена | | 784 593,88 |
| 6. НДС (12% от цены) | | 94 151,26 |
| 7. Цена реализации с НДС | | 984 194,57 |

### 5.6 Экономический эффект и эффективность

Ввиду проектно-исследовательского характера работы и отсутствия конкретных объектов внедрения приведены общие данные экономического эффекта внедрения ИИ управления технологическим процессом сортировки из публичных источников.

Ожидаемый экономический эффект внедрения ИИ управления технологическим процессом сортировки:

1) повышение эффективности бизнес-процессов, благодаря:

а) увеличению длительности полезной работы оборудования;

б) увеличению межремонтного периода оборудования на 5-20% за счет оптимизации принимаемых решений, выбора оптимального вида;

2) повышение управляемости бизнес-процессов, благодаря:

а) уменьшению штата сотрудников, и следовательно – уменьшению зарплатного фонда;

2) также из-за сокращения штата – расходы на коммунальные услуги также уменьшатся, и также уменьшается риск получения сотрудниками производственных травм;

2) автоматизированному формированию ведомости капитального ремонта. Система будет самостоятельно определять уровень необходимого ремонта;

3) уменьшению временных затрат оператора с 5 чел/час до 1 чел/час.

Становится возможным регулярный и оперативный анализ данных, вносимых на сервер хранения данных.

4) снижению экономических потерь на 5% в год за счет снижения аварийности, повышения надежности, что, в свою очередь, связано с анализом и прогнозированием состояния оборудования, своевременной разработкой и реализацией предупреждающих мероприятий.

5) увеличения экологичности производства за счет малого энергопотребления узла сортировки.

## Вывод

В данном разделе было проведено экономическое обоснование дипломного проекта был проведен расчет трудоемкости разработки системы, на основе которой был создан и проанализирован сетевой график. Были рассчитаны затраты на разработку проекта: затраты на оплату труда, амортизационные затраты, затраты на текущий и профилактический ремонт, затраты на материалы и затраты на оплату потребляемой электроэнергии. Также были рассчитаны предполагаемая прибыль и розничная цена на поставку с учетом НДС.

Так как все затраты включены в цену поставки, то для данного проекта нет необходимости рассчитывать величину инвестиций и срок окупаемости проекта. И оценивание экономического эффекта от внедрения системы пройдет только через неопределенный период, после которого, основываясь на отчетности за этот самый период, можно будет судить об улучшении экономического эффекта и соответствует ли он ожидаемому. Но в данный момент можно гарантировать, что система привнесет значительный вклад в расширение деятельности предприятия посредством той же бесперебойной работы узла сортировки.

## Заключение

В данном дипломном проекте, были произведены исследование и разработка нейронной сети глубокого обучения для технологического процесса сортировки деталей.

В первой, технологической части были выполнены 3 из четырех главных задач, поставленных в начале создания проекта:

1) были изучены теоретические аспекты разработки интеллектуальных систем глубокого обучения. Именно благодаря методу глубокого изучения, было исследовано и приведено множество положительных аспектов, влияющих на процент ошибок и отклонений результата, а также, именно глубокое обучение позволяет увеличить эффективность и результативность разработанной нейронной сети;

2) опираясь на множество научной литературы, был проведен развернутый анализ математической модели архитектуры нейронной сети, которая основывается на сложных математических, алгебраических понятий и вычислений, которые также включают в себя принципы теории неопределенности, которые применяются при вычислении и последующем нахождении множества конечных результатов. Исходя из чего прогноз результата работы нейронной сети может быть максимально близок к реальному результату;

3) было проведено исследование микрокомпьютеров, их архитектуры и характеристик, из которого последовало обоснование выбора именно одноплатного микрокомпьютера Raspberry Pi 3B+, были изучены все положительные и негативные стороны его использования в качестве аппаратного обеспечения системы;

В дальнейших же двух частях: конструкторской и программной, была рассмотрена и выполнена главная задача проекта – разработка нейронной сети распознавания объектов. Именно эта задача была поставлена во главу угла, и изучена так подробно, насколько того позволяли условия. Ведь только проведя исследование данной темы, её проблематики, сложности, правил, которым подчиняется модель, можно было приступать к разработке системы.

Сама же система, как с экономической, так и со стороны соблюдения правил безопасности жизнедеятельности, в результате расчетов доказала свою актуальность, применимость во многих отраслях, гибкость во внедрении на производства и безопасность при работе.

Для дальнейшего усовершенствования системы планируется внедрение манипуляторов, что могло бы также открыть возможность применения системы для управления технологическими процессами, как и сортировки, так и дальнейшей сборки деталей. Именно доказанная актуальность и новизна проекта позволяют выстраивать перспективы её улучшения.

## Список литературы


1. Орельен Жерон, Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow, Компьютерное издательство "Диалектика", 2018 г;

2. Шолле Франсуа, Глубокое обучение на Python. — СПб.: Питер, 2018 г. — 400 с.: ил. — (Серия «Библиотека программиста»);

3. Андрей Бурков, The Hundred-Page Machine Learning Book, - свободно распространяемая литература;

4. Адриан Роузброк, Глубокое обучение для компьютерного зрения в среде программирования Python, https://www.pyimagesearch.com/deep-learning-computer-visionpython-book/ PYIMAGESEARCH: 2017 г.;

5. М. Рафикззаман, Цифровая логика и дизайн микрокомпьютеров, WILEY-INTERSCIENCE, 2005 г., Хобокен, Нью-Джерси;

6. Андрю В. Траск, Grokking Deep Learning, Manning Publications, 2017 г.

7. Ашвин Паджанкар, Raspberry Pi Computer Vision Programming (free sample), PACKT publishing, open resource;

8. Кристофер М. Бишоп, Pattern Recognition and Machine Learning. Springer Science+Business Media 2006 г., Сингапур;

9. Чару С. Аггарвал, Нейронные сети и глубокое обучение, Springer International Publishing AG, 2018 г., Швецария;

10. Ричард Шелиски, Computer Vision, Springer-Verlag London Limited 2011 г., British Library

11. Глубокое обучение и Raspberry PI, интернет-ресурс https://habr.com/ru/post/400141/

12. Network-in-Network trained in Torch7, интернет-ресурс https://gist.github.com/szagoruyko/0f5b4c5e2d2b18472854

13. Обнаружение объектов с помощью Глубокого Обучения на Raspberry Pi, интернет-ресурс https://apptractor.ru/develop/obnaruzhenie-obektov-s-pomoshhyu-glubokogo-obucheniya-na-raspberry-pi.html

14. Боканова Г.Ш., Методические указания по выполнению экономической части выпускной работы – Алматы: АУЭС, 2020 – 26 с.

15. Липаев В. В. Технико-экономическое обоснование проектов сложных программных систем. — М.: СИНТЕГ, 2014, - 284 с.

16. ГОСТ 34.003-90 Информационная технология. Автоматизированные системы. Термины и определения.

17. Сейдаметова З.С., ЭКОНОМИКА И МАШИННОЕ ОБУЧЕНИЕ - научная статья, 2019, - 171 с.

18. Кент Бек, Экстремальное программирование. Разработка через тестирование – 2000 – 260 с

19.project.dovidnyk.info/index.php/home/upravlenieproektamiposozdaniyuprogrammnogoobespecheniya/po – «Управление проектами по созданию программного обеспечения. Эволюция экономики разработки.»

20. Н.Г. Приходько, С.Е. Мананбаева, Ж.М. Айтбаева. Дипломное проектирование. Методические указания для студентов специальности 5В073100. – Алматы: АУЭС, 2014. – 23 с.

21. Трудовой кодекс РК (с изменениями и дополнениями по состоянию на 13.05.2020 г

22. Закон Республики Казахстан от 28 февраля 2004 года № 528-II О безопасности и охране труда

23. Приказ Об утверждении Санитарных правил "Санитарно-эпидемиологические требования к объектам промышленности" от 20 марта 2015 года № 236

24. Дубовцев В.А. Безопасность жизнедеятельности. / Учеб. пособие для дипломников. - Киров: изд. КирПИ, 1992.

25. Безопасность жизнедеятельности. /Под ред. Н.А. Белова - М.: Знание, 2000 - 364с

26. Самгин Э.Б. Освещение рабочих мест. – М.: МИРЭА, 1989. – 186с.