МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет (НИУ)»
«Институт естественных и точных наук»
Физический факультет
Кафедра физики наноразмерных систем

РАБОТА ПРОВЕРЕНА

Рецензент, к. ф-м н., доц.

_____/Клебанов И. И./

«_____» июня 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой д ф-м н. проф.

_____/Воронцов А. Г./

«____» июня 2020 г.

Разработка системы распознавания жестов с использованием модели машинного обучения

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 11.04.04.2020.263 ПЗ ВКР

Руководитель, к. ф-м н., доцент

_____/Подгорнов Ф. В./

«____»_____ 2020г.

Автор, студент группы ЕТ – 263

_____/Горобец А. В./

«____»_____ 2020г.

Нормоконтролер, к.т.н., доцент

_____/Колмакова Н. С./

«____»_____ 2020г.

Челябинск 2020

АННОТАЦИЯ

Горобец А. В. Разработка системы распознавания жестов с использованием модели машинного обучения. – Челябинск: ЮУрГУ, ИЕТН, ЕТ-263; 2020, 89 с. 25 рис., 6 табл., библиогр. список – 42 наименования.

Ключевые слова: Распознавание жестов, интернет вещей, МЭМС, беспроводные технологии передачи данных, машинное обучения, сверточная нейронная сеть, полносвязная нейронная сеть, решающие деревья.

В настоящее время интерес к технологии захвата движений находится в таких областях как игровая индустрия, компьютерная анимация, робототехника и др.

Захват движения – это технология для записи движений, которые затем можно использовать как альтернативный способ управления объектами (играми, роботами, компьютером). Тело человека имеет сложное строение, поэтому записывать его движения гораздо проще, чем создавать трехмерные модели, которые необходимо анимировать вручную.

В работе описан принцип создания системы для распознавания жестов, представлена принципиальная схема устройства на базе Arduino, а также архитектура используемой нейронной сети. В ходе работы был использован принципиально новый принцип захвата движений, основанный на вычислении средневзвешенного показания нейронной сети. Были проведены эксперименты с различными моделями машинного обучения, результаты этих экспериментов приведены в третьей главе.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время интерес к технологии захвата движений находится в таких областях как игровая индустрия, компьютерная анимация, робототехника и др.

Захват движения – это технология для записи движений, которые затем можно использовать как альтернативный способ управления объектами (играми, роботами, компьютером). Тело человека имеет сложное строение, поэтому записывать его движения гораздо проще, чем создавать трехмерные модели, которые необходимо анимировать вручную.

Существует два принципиальных подхода к захвату движений:

1) анализ видеопотока;

2) подготовка данных о движении датчиками на местах крепления

Целью данной работы является проверка гипотезы о возможности создания простого гаджета для распознавания движений рук, а также создание опытного образца.

Задачи работы:

1. Провести обзор литературы на тему современных технологий изготовления IoT датчиков, современных технологий передачи данных, а также современных методов обработки больших объемов информации;
2. Выбрать компонентную базу для создания опытного образца и собрать его;
3. Разработать программу прошивки для опытного образца;
4. Провести эксперименты с различными моделями машинного обучения.

Актуальность данной работы обусловлена быстрыми темпами роста рынка устройств для «умного города» и «умного дома», так по данным Discovery Research Group от 17 января 2020 года объем рынка систем «умного дома» в России в 2019 году вырос на 16,5 % и достиг 10,5 миллиардов рублей[1]. Также на основе нашей технологии возможно создание продукта для распознавания азбуки жестов глухонемых. В настоящее время это до сих пор остается

актуальной задачей, поскольку появление подобного устройства станет первым шаг в масштабной социализации глухонемых людей. Разработка таких устройств в последние годы ведется многими группами, в том числе и в России. Можно выделить некоторые из них:

- Облачная платформа синхронного перевода speakus. Главным недостатком этой платформы является то, что она не работает в онлайн режиме. То есть пользователь записывает видео с докладом на языке жестов, отправляет на платформу, через 24 часа получает озвучку для этого видео, и наоборот;
- Устройство на основе электромиографии. Недостатками технологии является ее высокая цена, нестабильная работа из-за использования программного обеспечения на основе классических алгоритмов, а также то, что пользователю предлагается достаточно габаритный девайс для руки;
- Яндекс.Разговор – является своего рода чатом для общения глухонемых и слышащих людей. Глухонемой пользователь печатает текст, на устройстве слышащего текст воспроизводится в аудио-формате. Слышащий человек записывает голосовое сообщение, которое на устройстве глухонемого воспроизводится в текстовом формате.
- Приложение распознавания жестов по видео от компании DeafSkills. Ограничениями данного продукта являются: обязательное нахождение человека в области видимости управляющего устройства, стабильность работы зависит от освещенности помещения, нестабильная работа из-за различных антропометрических данных пользователей.

Стоит также упомянуть, что ООН обратили внимание на ситуацию социализации глухонемых людей в Российской Федерации. В июле 2018 года в Москве в представительстве ООН состоялась презентация проекта DeafSkills, целью которого является помощь в трудоустройстве и социальной адаптивности молодым людям с нарушением слуха. Данный проект получил грант от Фонда Демократии ООН.

DeafSkills стремится к расширению географии образовательных учреждений и созданию образовательных программ для глухих. По их мнению это даст возможность талантливым молодым людям, с нарушениями работы

слуха проявить себя и использовать в полной мере весь их трудовой, творческий и интеллектуальный потенциал для развития регионов в частности и страны в целом.

По мнению инициаторов этого проекта глухих людей должны принимать во все учебные заведения, не ограничивая при этом их выбор и создавая условия для их обучения. Государство и правительственные органы должны понимать, что подобные вложения окупятся, так как вклад в человеческий потенциал всегда конвертируется в виде будущих налогоплательщиков, что в свою очередь повлечет вклад в развитие экономики страны.

ЛИТЕРАТУРНЫЙ ОБЗОР

1.1. Микроэлектромеханические системы.

Как следует из названия, микроэлектромеханические системы (МЭМС) — это технология, которая сочетает в себе электрические и механические системы в микромасштабе. Практически любое устройство, изготовленное с использованием методов фотолитографии в микрометрическом масштабе, которое использует как электрические, так и функциональные возможности можно было бы считать МЭМС.

Микросистемы, которые принято обозначать аббревиатурой MEMS (MicroElectroMechanical-Systems), представляют собой миллиметровые/субмиллиметровые устройства, реализующие определенную функцию трансдукции между двумя (или более) различными физическими величинами, среди которых механические всегда участвуют. Проще говоря, независимо от конкретной функции, для которой он задуман, устройство МЭМС всегда имеет структурные части, которые перемещаются, изгибаются, растягиваются, деформируются и/или контактируют друг с другом. Эти особенности делают микросистемные устройства особенно подходящими для реализации очень широкого спектра микроразмерных датчиков и исполнительных механизмов[2].

Эволюционировавшая от технологий изготовления полупроводников, самая поразительная особенность технологии МЭМС заключается в том, что она позволяет создавать движущиеся микроструктуры на подложке. С этой возможностью можно создавать чрезвычайно сложные механические и электрические системы. Массы, изгибы, приводы, детекторы, рычаги, рычажные соединения, шестерни, амортизаторы и многие другие функциональные строительные блоки могут быть объединены для построения сложных систем на кристалле. Инерциальные датчики, такие как акселерометры и гироскопы, используют эту возможность в полной мере.

МЭМС являются исключительно пассивными устройствами, в отличии от транзисторов. Однако транзисторы не имеют подвижной или деформируемой части, то есть они никогда не используют механическую/структурную область для реализации функций трансдукции.

Развитие и МЭМС, и транзисторов следует концепции миниатюризации. Однако, если полупроводниковые устройства (транзисторы), помимо понижающего масштабирования, реализуют в электронной области мультифизическую функцию традиционных компонентов, МЭМС часто миниатюризируют классические объекты, сохраняя их трансдукцию через физические величины.

Однако, в то время как КМОП-транзисторы, сегодня находятся в диапазоне десятков нанометров, МЭМС может охватывать от нескольких микрометров (размеры в плоскости) до сотен микрометров или даже до нескольких миллиметров.

По мере того, как индустрия МЭМС развивается, стоимость производства корпуса становится лимитирующей составляющей конечной стоимости таких продуктов, как микроакселерометры и микрогироскопы [3]. Недорогие, но герметичные схемы упаковки, совместимые с хрупкими механическими компонентами, в настоящее время недоступны. Обработка на уровне пластин и чипов для МЭМС сопряжена с большим количеством осложнений, чем в обычных ИС. Необходимо проявлять крайнюю осторожность при выборе инструментов и условий окружающей среды (например, влажность и загрязнение частицами), поскольку свободные механические элементы могут подвергаться воздействию окружающей среды.

1.1.1 Применение МЭМС датчиков.

Чтобы завершить обзор технологий микросистем в общем плане, и прежде, чем приступить к углубленному обсуждению микро- акселерометров и гироскопов, будет кратко представлено несколько ссылок и примеров использования датчиков и исполнительных механизмов МЭМС на рынке.

Во-первых, следует подчеркнуть, что если микросистемные технологии начали развиваться в 1970-х годах, то первый коммерческий продукт на основе МЭМС, а именно поверхностный микромеханический акселерометр[4], появился только в начале 1990-х годов, почти два десятилетия спустя. Обоснованием длительного времени выхода на рынок, является сложная конструкция таких устройств, состоящих из многочисленных плиток, отличающихся такими ключевыми моментами, как надежность[5], упаковка, интеграция, а также затраты и, наконец, не в последнюю очередь, готовность рынка.

Начиная с первого коммерческого акселерометра, упомянутого в [4], МЭМС-датчики и исполнительные механизмы начали распространяться в различные системы и устройства, обеспечивая их присутствие во многих сегментах рынка. Сами акселерометры, например, стали в 1990-е годы стандартом де-факто в автомобильном секторе, применяясь в качестве датчиков замедления, активирующих надувание подушек безопасности в случае автомобильных аварий[6].

В последнее десятилетие, с появлением домашних игровых консолей, взаимодействующих с движением человека, а затем с массовым распространением смартфонов и планшетов, основанные на МЭМС инерциальные измерительные блоки стали стандартными компонентами, предоставляемыми широким кругом производителей оригинального оборудования[7].

Говоря уже о других классах устройств, МЭМС распространяются в коммутации/мультиплексировании оптических сигналов (исполнительные механизмы), миниатюрных микрофонах, датчиках давления/газа/температуры, тензометрических датчиках/ датчиках деформации и т. д. Целевые области применения весьма разнообразны, они варьируются от автомобильной до бытовой электроники, а также от космической отрасли/ ВПК до медицины.

В конечном счете, можно с определенной уверенностью утверждать, что тенденции в эксплуатации МЭМС будут усиливаться и в последующие годы.

Важнейшие рынки – это Интернет вещей (IoT) [8] и Интернета всего (IoE), на которых пользователи требуют наличия меньших, более дешевых, менее энергозатратных, многофункциональных и более специализированных датчиков и исполнительных механизмов, которые будут интегрироваться во все большее число умных городов, зданий, устройств, заводов, автомобилей, объектов.

1.1.2. Принципы поверхностной микрообработки.

Обзорная статья Петерсена в 1982 году [9] привела к широкому признанию микромеханического кремния в качестве конструкционного материала. В статье были обобщены основные механические свойства монокристаллического кремния, а также приведен ряд примеров структур, иллюстрирующих потенциал МЭМС. Кремний в поликристаллической тонкопленочной форме обладает свойствами (например, модулем Юнга, прочностью на разрыв и др.), которые обычно отличаются от значений монокристаллического кремния.

Ранние применения МЭМС установили, что монокристаллический кремний является прочным механическим материалом; обладает анизотропией модуля Юнга и близок к модулю нержавеющей стали, и он имеет твердость по Кнупу (по отпечатку пирамиды) в два раза больше, чем у железа, и очень высокую прочность на растяжение. Монокристаллический кремний, тем не менее, хрупок и ломается, не поддаваясь на излом [9]. Тонкопленочный поликристаллический кремний сохраняет некоторые из этих свойств, за некоторыми исключениями. Регулярно наблюдались более низкие сравнительные значения модуля Юнга и неупругой деформации. Поликристаллический кремний показал более больший разброс в распределении напряжений разрушения, чем монокристаллический кремний, где характеристики разрушения определяются геометрическими дефектами, такими как микродефекты[10]. Поликристаллический кремний может быть не столь чувствителен к этим микрогеометрическим неоднородностям.

Появление поликремния в качестве механического материала в 1980-х годах привело к значительным исследованиям и производственным разработкам

его механическим характеристикам в лабораториях и компаниях по всему миру. В то время как технолог МЭМС в первую очередь интересуется свойствами материала, такими как средняя деформация, градиент деформации, модуль Юнга, прочность на разрыв и демпфирование материала, он должен иметь в виду, что эти параметры являются функциями морфологии пленки, которая, в свою очередь, зависит от особенностей осаждения и/или обработки изготовления.

Поверхностная микрообработка подразумевает последовательное выращивание слоев на кремнии. Типичным процессом является нанесение на поверхность пластины тонких пленок, затем формирование методом фотолитографии защитного рисунка и последующее травление пленки. Так как в МЭМС основная задача сформировать подвижные функционирующие механизмы, в слоях чередуются пленки конструкционного материала (Si), и абляционного материала (обычно это SiO_2). Сами функциональные элементы образуются из конструкционного материала. Абляционный материал вводится с целью заполнения пустоты между подвижными элементами. Последней технологической операцией является удаление травлением абляционного материала, благодаря чему конструкционные элементы приобретают требуемую подвижность.

В случае классического технологического процесса: конструкционный материал – кремний, абляционный – двуокись кремния, последняя операция представляет собой погружение пластины в плавиковую кислоту (раствор HF), который селективно взаимодействует с двуокисью кремния, при этом не взаимодействую с самим кремнием.

Затем, пластины разрезают на отдельные кристаллы, которые в последствии упаковываются в корпусы, с заданными параметры для эксплуатации в тех или иных условиях.

В отличии от объемной микрообработки, поверхностная требует большего количества технологических операций, поэтому как следствие она дороже. С

целью минимизации затрат поверхностная микрообработка используется в том случае, когда требуется создать сложные механические элементы[10].

1.1.3. МЭМС акселерометры.

Первыми инерциальными датчиками, изготовленными на кремнии, как говорилось выше, были акселерометры. Многие из этих устройств были и остаются произведенными на основе массивного груза и пружины. Простой микроакселерометр может быть сделан в виде пружинного маятника и груза, либо шариков в катушке[10].

Первые кремниевые устройства были основаны на объемных микромеханизмах, используя пьезоэлектрическое считывание. Такие устройства также в литературе называют пьезорезистивными датчиками ускорения

Особенностью этого устройства являлся самотестируемый электрод. Это позволяет сенсорам быть протестированными на предмет нормальной работы системы. Когда устройство активировано, самотестируемый электрод, за счет электростатической силы тянет массивный грузик вверх к себе. Затем с пьезорезисторов считываются показания и основываясь на этих показаниях контроллер проверяет, что масса движется, как требуется[10].

Первоначально для изготовления этих устройств использовались однотипные процессы объемной микрообработки как и при изготовлении датчиков давления[11]. На данный момент совместно с объемной микрообработкой используется и поверхностная.

Пьезорезистивный датчик ускорения состоит из тонкой лопасти, соединенной с опорной рамкой, как показано на рисунке 1.1. Лопатка способна свободно двигаться, реагируя на ускорение. Пьезорезисторы находятся вблизи места крепления лопатки. По мере того как лопатка отклоняется, величина электрического сопротивления изменяется в ответ на возникающее механическое напряжение[11].

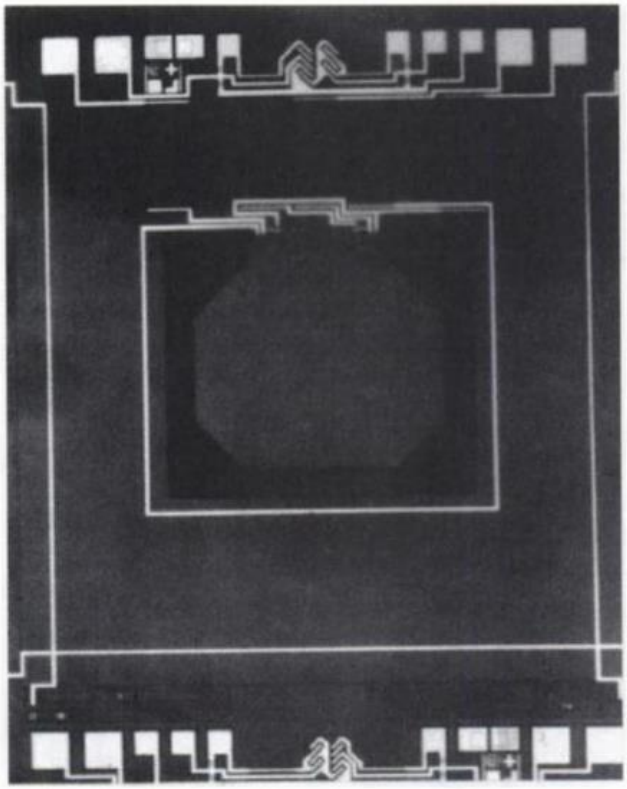

Рисунок 1.1. Фотография пьезорезестивного акселерометра.

Акселерометры этого типа используются в двух системах пассивно-сдерживающего контроля. Во-первых, акселерометр сопрягается с интегральной схемой формирования сигнала и температурной компенсации и собирается в пользовательском корпусе, пригодном для монтажа на печатной плате. После того, как акселерометр в сборе протестирован и откалиброван в температурном диапазоне, он собирается в модуль управления пассивным ограничителем бокового удара. Модуль управления содержит акселерометр, микроконтроллер и программный алгоритм.

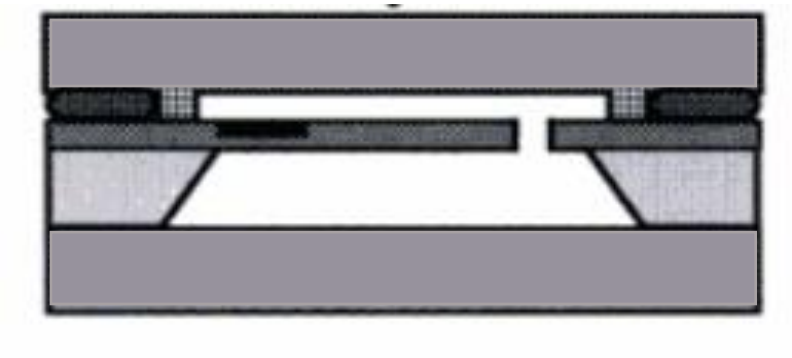

Рисунок 1.2. Структурное представление рабочей области пьезорезестивного акселерометра.

Масса микромеханизма и консольные размеры являются в первую очередь основными конструктивными параметрами. Второстепенными, но важными вопросами проектирования микроакселлерометров являются демпфирование, особенно для микроструктур с большой площадью и резонансная частота устройства. Избыточная чувствительность может привести к ошибке, когда нежелательные ускорения обнаруживаются с неправильных направлений. Для промышленных, аэрокосмических и автомобильных применений вибрации системы ниже 2000 Гц являются общими. Резонансная частота микроструктуры должна быть выше этого значения или использовать обширное пакетное демпфирование, чтобы резонансные колебания не интерпретировались как ложные ускорения[12].

Наконец, при проектировании акселерометра следует учитывать напряжение пленки на тонкой лопасти или в нем, а также наличие свободных частиц. Если напряжение пленки достаточно велико, лопасть может отклоняться вверх или вниз в состоянии покоя и касаться границ рабочей области. Эти контакты, как и присутствие частиц, может затруднить движение и снова привести к ошибке.

1.1.4. МЭМС гироскопы.

Макроскопические гироскопы используют вращающееся колесо, прикрепленное к карданной конструкции, однако вращающиеся колесные гироскопы имели много недостатков, в первую очередь связанных с трением и износом подшипников, вибрирующие гироскопы, такие как полусферический резонаторный гироскоп и камертонные гироскопы представляют собой эффективное решение проблем подшипников путем исключения вращающихся частей.

Были также разработаны альтернативные высокоэффективные технологии, такие как волоконно-оптический гироскоп (FOG) и кольцевой лазерный гироскоп (RLG). Устраняя практически все механические ограничения, такие как чувствительность к вибрации, ударам и трению, эти оптические гироскопы нашли много высококлассных применений, несмотря на их высокую стоимость.

Несмотря на широкое разнообразие конструкций и принципов работы микромеханических гироскопов, большинство из представленных микромеханических гироскопов используют вибрирующие механические элементы для определения угловой скорости. Это основная причина, по которой вибрационные гироскопы были успешно миниатюризированы с использованием процессов микрообработки и стали привлекательной альтернативой своим макромасштабным аналогам[13].

Основной принцип работы микромеханических вибрационных гироскопов основан на синусоидальной силе Кориолиса. Масса обычно подвешивается над стеклянной подложкой с помощью подвесной системы, состоящей из гибких балок. Общая динамическая система, как правило, представляет собой систему масс-пружин-демпферов с двумя степенями свободы (2-DOF). Такие устройства имеют вертикальную структуру.

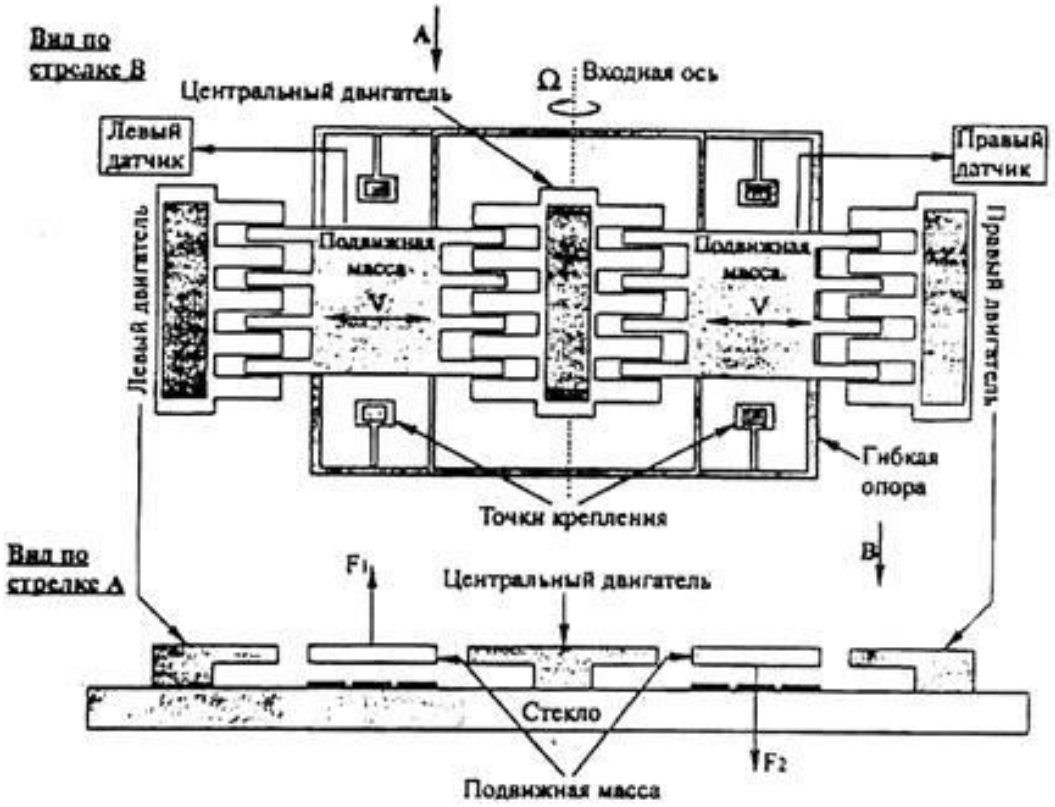

Рисунок 1.3. Конструкция вибрационного гироскопа.

Гироскоп имеет два подвижных груза (подвижные массы) (рисунок 1.3), закрепленных на гибких балках относительно стеклянной подложки. Эти грузы приводятся в вибрационное движение посредством трех электростатических двигателей со скоростью V, параллельной к плоскости подложки. Векторы этих скоростей противоположно направлены друг к другу. В случае поворота гироскопа, с угловой скоростью Ω, появляется силы Кориолиса F1 и F2, которые будут стремиться вернуть систему в состояние равновесия. Под действием этих сил, один груз начнет подниматься, а другой наоборот опускаться относительно плоскости вибрации [14]. Значение вышеупомянутых сил можно оценить по формуле (1):

$$F_{\text{к}} = 2V\Omega m, \tag{1}$$

где m – масса подвижного элемента.

Эти перемещения грузов зафиксируют емкостные датчики, которые находятся на подвижных элементах и подложке, и сформируют выходной сигнал гироскопа, который в последствие обрабатывается АЦП.

1.1.5. Выводы по разделу.

Были изучены современные технологии производства микро-акселерометров и гироскопов, а также перспективы их применения в гаджетах IoT и автомобильной промышленности.

Стоит отметить, что из-за сложной конструкции таких приборов надежность их работы зависит от таких внешних факторов, как фоновое излучение, вибрационный режим, влажность, температура и т.д. Поэтому на этапе проектирования специалист, работающий с МЭМС, обязательно должен изучить условия, в которых планируется эксплуатировать прибор, и основываясь на этих данных, подобрать компонентную базу для надежной работы конечного продукта.

1.2.Беспроводные технологии передачи данных.

Выделяют 4 основных класса беспроводных технологий[15]:

1) Беспроводные персональные сети. Пример: Bluetooth;
2) Беспроводные локальные сети. Пример: Wi-Fi;
3) Беспроводные сети масштаба города. Пример: WiMAX;
4) Беспроводные глобальные сети. Пример: LTE.

Bluetooth — производственная спецификация беспроводных персональных сетей. Данная технология позволяет передавать данные между персональными устройствами (ПК, смартфоны, принтеры, наушники, джойстики и т.д.) на доступной радиочастоте для ближней связи. Старые версии спецификации позволяли устройствам взаимодействовать на расстоянии около 10 метров, современные – на более, чем 1500 метров. Дальность зависит от наличия преград и помех, даже в масштабах одного помещения[16].

Wi-Fi — технология беспроводной локальной сети на основе стандартов IEEE 802.11. Как правило сети Wi-Fi содержат не менее одного клиента и не менее одного сервера. Также протоколы Wi-Fi позволяют подключиться двух клиентов друг к другу в режиме «точка-точка». То есть точка доступа (сервер) не используется, а клиенты подключаются друг к другу напрямую для передачи

данных. Более подробно принцип работы описан в официальном тексте этого стандарта, однако стоит отметить, что этот документ не описывает всех аспектов построения сетей на этой технологии. Ввиду этого производители оборудования вольны сами выбирать какие подходы применять при решении той или иной задачи[17].

WiMAX – технология для предоставления беспроводной связи на больших масштабах. Технология WiMAX решает следующие задачи:

- Соединения точек доступа (серверов) Wi-Fi в единую сеть и с глобальной сетью Интернет;
- Обеспечение беспроводного широкополосного канала передачи данных, что на данный момент на территории РФ делают сети DSL;
- Создание точек доступа (серверов), не локализованных географически;
- Создание систем удаленного мониторинга и интеграция их в единую систему («умный город»).

WiMAX предоставляет пользователям доступ в Интернет на высокой скоростью, как и Wi-Fi, но в отличии от последнего обладает большей площадью покрытия. Это позволяет использовать данный стандарт связи в качестве «магистральных каналов», что в будущем потенциально может вытеснить традиционные DSL и крупные локальные сети. Обобщая все вышесказанное, можно подытожить, что используя подобный подход можно создавать масштабируемые высокоскоростные сети в рамках города и даже муниципальных округов[18].

LTE (часто обозначается как 4G LTE) — современный стандарт беспроводной высокоскоростной передачи данных для доступа к глобальной сети Интернет. Он основан на сетевых технологиях GSM/EDGE и UMTS/HSPA, увеличивая пропускную способность и скорость за счёт использования другого радиоинтерфейса вместе с улучшением ядра сети, в отличие от предыдущего поколения 3G[19].

Так как в нашей работе нас интересуют в первую очередь персональные сети, далее мы более подробно остановимся на технологии Bluetooth.

Рассмотрим принципы работы этого стандарта связи, основные протоколы работы, а также правило формирования сообщений Bluetooth устройств.

1.2.1. Bluetooth.

Bluetooth является бесплатной спецификацией передачи данных. Ключевой особенностью спецификации Bluetooth является то, что она разрешает многим различным производителям работать друг с другом. С этой целью Bluetooth не просто определяет физический уровень радиосистемы, а также регламентирует программный стек для того, чтобы устройства, работающие на этой технологии могли сопрягаться друг с другом, а также передавать данные для обмена сообщениями, управления и т.д.[20].

Bluetooth Special Interest Group (SIG) - это группа компаний, работающих вместе с целью продвигать и определять спецификацию Bluetooth. Bluetooth SIG была основана в феврале 1998 года следующими компаниями:

1. Ericsson Mobile Communications AB;
2. Intel Corp.;
3. IBM Corp.;
4. Toshiba Corp.;
5. Nokia Mobile Phones;

В мае 1998 года основные промоутеры публично объявили о создании global SIG и пригласили другие компании присоединятся к SIG в качестве компаний, поддерживающих Bluetooth в обмен на обязательную поддержку спецификации Bluetooth. Основные промоутеры опубликовали версию 1.0 спецификация Bluetooth в июле 1999 года на веб-сайте Bluetooth, http://www.bluetouth.com [21].

1.2.2. Стек протоколов Bluetooth.

Весь стек протоколов Bluetooth можно разделить на три большие части, как показано на рисунке 1.4[22]:

1) Протоколы контроллера;
2) Протоколы хоста;
3) Протоколы приложений.

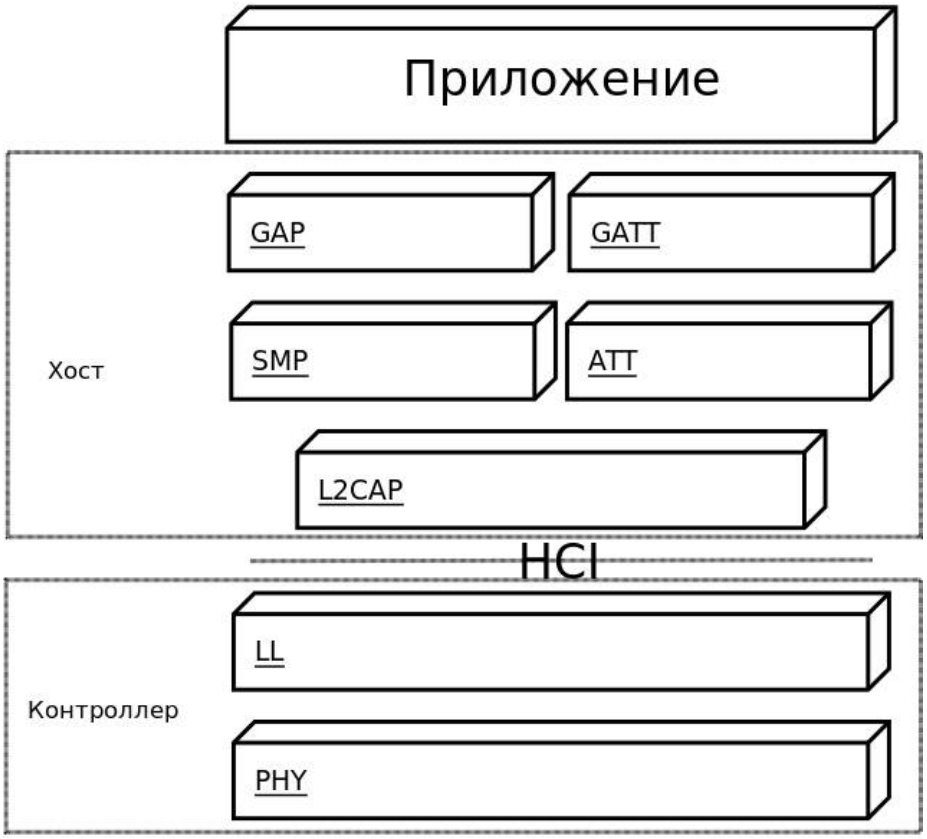

Рисунок 1.4. Упрощенное представление стека протоколов Bluetooth.

Протоколы контроллера – это самые низкий уровень протоколов, которые определяют работу Bluetooth устройств на физическом уровне[22]:

- PHY – объединяет в себе два протокола, которые обеспечивают связь между двумя Bluetooth устройствами. RF – радиоканал Bluetooth, BB – определяет правила построения сообщений в базовой полосе частот на низшим уровне, а именно количество байт в одном сообщении и назначения каждого байта под определенную задачу;
- LL или LMP – протокол администратора канала. Этот протокол определяет правила построения коммуникации Bluetooth модуля с обвязкой и прочей периферией. Основными протоколами для этих целей являются сторонние общепринятые высокоуровневые протоколы I2C и SPP.

Протоколы хоста – это протоколы, определяющие правила передачи информации между двумя Bluetooth устройствами в том или ином случае[22]:

- L2CAP – Протокол управления логическим каналом и адаптацией. Этот протокол расположен сразу над физическим уровнем и обеспечивает следующие функции:
    - мультиплексирование протоколов;
    - сегментацию;
    - реассемблирование;
    - проведение обмена информацией относительно качества передачи;
    - групповое управление.
- SMP – с помощью этого протокола Bluetooth устройства обнаруживает доступные услуги, а также их характеристики и свойства. Под услугами здесь понимается доступ к поисковой связи, точками доступа, другим доступным для сопряжения устройствам, сетевым мостам и т.д.;
- ATT – Протокол атрибутов. Это оптимизированный протокол, созданный специально для BLE (Bluetooth Low Energy). Принцип этого протокола в том, чтобы передавать минимально возможное количество байт;
- GATT – Общий протокол атрибутов. Этот протокол определяет правила составления сообщений стандартизованной длины 48 байт, где каждый байт содержит определенную информацию:
    - 2 байта – MAC адрес устройства;
    - 4 байта – преамбула;
    - 3 байта – UUID атрибута (описание сообщения, как необходимо его исполнять);
    - 38 байт – сообщение (полезная информация);
    - 1 байт – байт завершения передачи пакета.
- GAP – протокол типового профиля доступа. Формализует правила, по которым устанавливается и поддерживается связь между Bluetooth устройствами.

Отдельное место занимает протокол HCI. Этот протокол, который обеспечивает связь между протоколами контроллера и протоколами хоста.

Большее практическое значения имеют протоколы приложений, с которыми разработчик непосредственно взаимодействует:

- TCS (Telephony Control Protocol Specification) предоставляет услуги телефонии;

- WAP и OBEX – интерфейсы для взаимодействия с более высокоуровневыми коммуникационными протокола;
- RFCOMM обеспечивает последовательный интерфейс типа RS232. Имеет режим работы Transparent UART, то есть это протокол, который обеспечивает передачу информации между устройствами так, если бы они были соединены между собой кабельным каналом.

1.2.3. Физические основы передачи данных.

Устройства Bluetooth работают на частоте 2,4 ГГц в глобально доступной безлицензионной ISM полосе. Эта полоса зарезервирована для общего использования промышленными, научными и медицинскими организациями (ISM), которые подчиняются базовому набору мощности, спектрального излучения и помех[20].

Рабочая полоса разделена на каналы с интервалом в I МГц. Схемой модуляции сигнала является GFSK (Гауссовская манипуляция сдвигом частоты). Используя GFSK, двоичная 1 означает положительное отклонение от номинальной несущей частоты, в то время как двоичный ноль означает отрицательное отклонение частоты.

После каждого переданного пакета оба устройства перенастраивают свои приемопередатчики на другую частоту. Таким образом, устройства Bluetooth используют весь доступный диапазон ISM и, если передача нарушена помехами на одном из их каналов, повторная передача всегда будет осуществляться по другому каналу.

Каждая итерация передачи пакета длится 625 микросекунд, и, как правило, устройства меняют частоту один раз за цикл.

Существует три класса Bluetooth устройств, в зависимости от мощности, которые обеспечивают передачу информации на расстояние 10, 20 и 100 метров соответственно.

Так как устройства должны перейти на новые частоты после каждого переданного пакета, они все должны согласиться на последовательность частот, которые они будут использовать. Устройства Bluetooth могут работать в двух

режимах: master и slave. Именно master задает порядок частот, на которых будет происходить передача данных. Подчиненные устройства(slave) синхронизируются с ведущим устройством по времени начала работы и последовательности частот.

Каждое Bluetooth устройство имеет уникальный MAC адрес устройства и таймер. Используя спецификацию Bluetooth можно вычислить последовательность частот, зная MAC адрес и идентификатор таймера. Когда ведомые подключаются к ведущему устройству, master сообщает им адрес своего устройства и таймер работы. Затем ведущие устройства используют эту информацию для расчета. Таким образом slave устройства синхронизируют свою последовательность устройств с последовательностью master.

В дополнение к управлению последовательностью частот, master контролирует, когда устройствам разрешена передача данных ведущему устройству.

Помимо того, что спецификация Bluetooth разрешает соединения типа «точка - точка» рисунок 1.5, она также позволяет создание пикосетей.

Рисунок 1.5. Соединение типа «Точка-точка».

Спецификация Bluetooth регламентирует нахождение до 8-ми устройств в одной пикосети: 1 master и 7 slave (рисунок 1.6). При этом ведущее устройство определяет для ведомых не только последовательность используемых частот, но и также индивидуально выделенные частоты для каждого ведомого устройства, на которых оно может отправлять пакеты master’у.

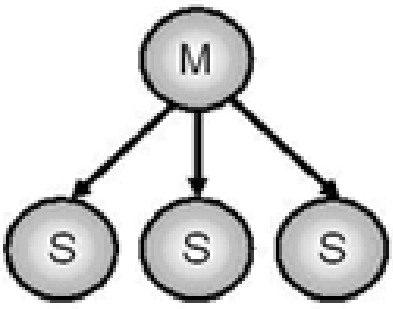

Рисунок 1.6. Принципиальная схема пикосети.

1.2.4. Выводы по разделу.

Были изучены современные технологии беспроводной передачи данных. Среди всего многообразия современных технологий был выбран стандарт Bluetooth, как наиболее подходящий для решения поставленной задачи.

Спецификация Bluetooth была нами изучена и представлена в виде краткого обзора в настоящем разделе на уровне достаточном для разработки устройств, использующих эту технологию для передачи данных.

Также кратко был затронут вопрос о возможности построения пикосетей. Подобные схемы соединения устройств в настоящей работе использоваться не будут, однако их внедрение потребуется при дальнейшей работе над проектом.

1.3.Машинное обучение.

Машинное обучение – это разновидность решения задач Искусственного интеллекта, при котором алгоритм не описывается напрямую, как в случае с императивными классическими алгоритмами, а подстраивание алгоритма под известные результаты сходных задач.

Используя такой подход, программист не пишет инструкцию, учитывая все возможные значения параметров, а лишь описывает алгоритм самостоятельного нахождения решения программой. При этом программа использует статистические данные, из которых выделяет закономерности, на основании которых в последствии делает прогнозы (предсказывает результат с той или иной долей вероятности).

Для того, чтобы научить программу решать ту или иную задачу, предварительно необходимо собрать и предъявить ей датасет(набор данных, включающий в себя параметры и известный результат). Именно этот датасет и будет обобщать модель машинного обучения. При этом датасет должен быть сбалансирован, иначе может произойти эффект переобучения.

Для построения моделей машинного обучения используются методы математической статистики, численных методов, теории графов и теории вероятности.

В этом разделе мы рассмотрим задачи, которые решает машинное обучения, наиболее распространенные модели машинного обучения, а также метрики, по которым можно сравнить две различных модели.

1.3.1. Типовые задачи машинного обучения.

Абсолютно любую задачу, которая решается с использованием машинного обучения можно отнести к одному из пяти типов[23]:

1. Регрессия – прогнозирование результата на основе различных значений фиксированного набора параметров. Результатом является вещественное число. К таким задачам относятся: прогнозирование прибыли организации в следующем квартале, стоимость автомобиля, продолжительность жизни индивида и др.;
2. Классификация – это задача группировки сущностей на основании набора признаков. Область допустимых значений ограничена. К таким задачам относятся: тегирование фотографий (описание фотографии раннее заданными словами), описание целевой аудитории по психотипу, установление стадии рака по снимкам МРТ и др.;
3. Кластеризация – автоматическое разделение сущностей на группы, при этом выделяя лимитирующие признаки для каждой группы. К таким задачам относятся: разделение населения РФ по покупательской способности и др.;
4. Уменьшение размерности – упрощение системы, посредством уменьшения числа признаков для дальнейшей визуализации и анализа. К таким задачам относятся: выделение наиболее важных параметров при проведении многофакторных экспериментов и др.;

5. Выявление аномалий – сепарация аномальных образцов от общей выборки. Может показаться, что эта задача похожа на задачу №2 «Классификация», однако стоит отметить, что, как правило, количество аномальных образцов для обучения стремится к нулю, то есть если мы будем решать задачу классификации, то получим несбалансированный датасет. К таким задачам относятся: выявление актов мошенничества с банковскими картами, поиск потенциального преступника по биографии и др.

В теории машинного обучения есть два основных подхода[23]:

1. Обучение с учителем – этот подход подразумевает, что при обучении модели ей представляют набор признаков и правильный ответ. То есть модель как бы должна запомнить правила, по которым необходимо сделать прогноз. Например, есть фотографии «кошек» и «собак», при этом мы загружаем в модель и фотографию, и описание того, какое животное на ней находится. Типичные задачи этого подхода – классификация и регрессия;
2. Обучение без учителя – этот подход подразумевает, что при обучении модели ей представляют датасет без описания, при этом задают на какое количество групп можно разбить этот набор данных. То есть модель должна сама выделить признаки, по которым происходит разделение на группы. Например, есть фотографии «кошек» и «собак», при этом мы загружаем в модель только фотографии и указываем, что данный датасет следует разделить на два класса. Типичные задачи этого подхода – кластеризация и уменьшение размерности.

Перед нами стоит задача классификации, поэтому далее в этом разделе пойдет речь о самых распространённых моделях решения подобных задач.

1.3.2. Линейная и логистическая регрессии.

Существует множество задач классификации, но логистическая и линейная регрессии являются самыми распространенными методами решения проблем бинарной классификации. Логистическая регрессия описывает и оценивает связь между одной зависимой двоичной переменной и независимыми переменными.

Логистическая регрессия может использоваться для различных задач классификации, таких как, например, обнаружение спама, предсказание у пациента диабета, будет ли пользователь нажимать на данную рекламную ссылку или нет и др.

Логистическая регрессия – это частный случай линейной регрессии, когда искомый результат носит категориальный характер. Этот алгоритм использует набор коэффициентов в качестве зависимой переменной. Логистическая регрессия предсказывает вероятность возникновения двоичного события с использованием функции (3).

Линейная функция описывается следующей функцией[24]:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n , \quad (2)$$

где y- искомая величина;

x_n – независимые переменные;

β_n – подбираемый коэффициент регрессии.

Функция, описывающая логистическую регрессию похожа на линейную и представляет собой следующее выражение:

$$y = 1/(1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}) \quad (3)$$

Область определения линейной регрессии носит непрерывный характер, в свою очередь область определения логистической регрессии можно считать квазидискретной. Коэффициенты линейной регрессии считаются с использованием обычного метода наименьших квадратов, а коэффициенты логистической регрессии с использованием метода оценки максимального правдоподобия[24].

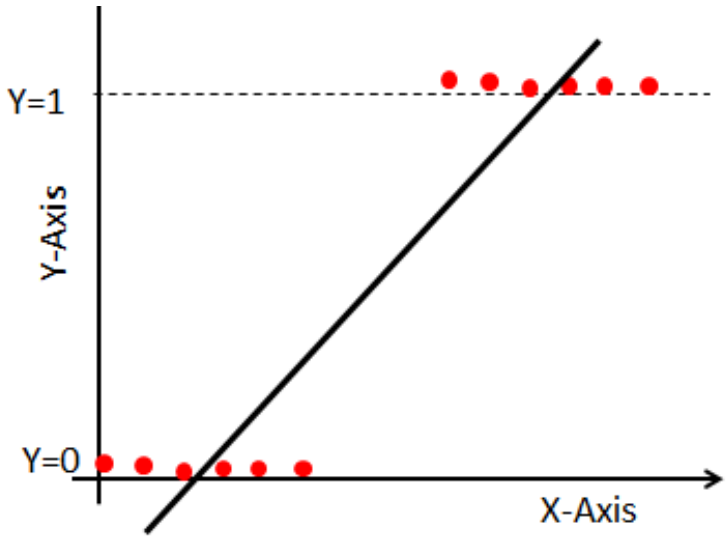

Рисунок 1.7. Пример применения линейной регрессии к задаче бинарной классификации.

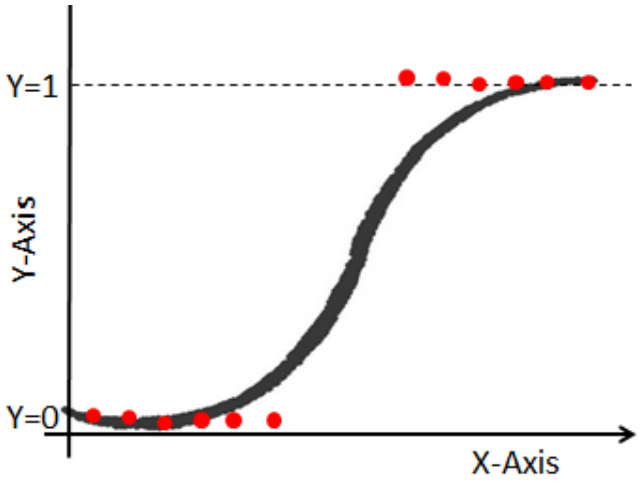

Рисунок 1.8. Пример применения логистической регрессии к задаче бинарной классификации.

Метод оценки максимального правдоподобия — это алгоритм при, котором происходит максимизация функции правдоподобия. Такой подход определяет коэффициенты, наиболее вероятные для получения наблюдаемых данных. С точки зрения статистики, метод оценки максимального правдоподобия устанавливает среднее и дисперсию в качестве параметров при определении конкретных коэффициентов для данной модели. Этот набор коэффициентов может быть использован для прогнозирования данных, необходимых при нормальном распределении.

В свою очередь при использовании метода наименьших квадратов коэффициенты вычисляют путем подгонки линии регрессии к экспериментальным точкам, при этом получают минимальную сумму квадратов отклонений. Оба этих метода используют для оценки коэффициентов линейной регрессионной модели. Метод оценки максимального правдоподобия предполагает совместную функцию вероятности для всех экспериментальных точек, в то время как метод наименьших квадратов не требует никаких стохастических допущений для минимизации расстояния.

Ниже описан пример обучения и применения логистической регрессии на языке python 3.5, с использованием библиотек pandas и sklearn (Рисунок 1.9).

```
import pandas as pd
from sklearn.cross-validation import train_test_split
from sklearn.linear-model import LogisticRegression

dataset = pd.read_csv("Горобец_диплом.csv", header=None)
feature_cols = ['feature1', 'feature2','feature3']
x = dataset[feature_cols]
y = dataset.lable

x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, random_state=0)

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
```

Рисунок 1.9. Пример программы для создания и обучения логистической регрессии.

Первый блок программы – подключение требуемых библиотек;

Второй блок – чтение датасета из csv файла;

Третий блок – разделение датасета на тренировочную и тестовую выборки в соотношении 3:1;

Четвертый блок – инициализация модели и ее обучение(расчет коэффициентов)(метод fit из библиотеки sklearn выполняет максимизацию функции правдоподобия);

Пятый блок – запуск обученной модели на тестовой выборке и запись результатов в массив y_pred.

1.3.3. Деревья принятия решений и случайный лес.

Дерево решений — это древовидная структура, похожая на блок-схему, в которой внутренний узел представляет объект (или атрибут), ветвь представляет правило принятия решений, а каждый конечный узел представляет результат. Самый верхний узел в дереве решений называется корневым узлом. Этот алгоритм обучаясь формирует ветви (правила). Дерево при обучении, развивается рекурсивно. Пример структуры дерева принятия решений показана на рисунке 1.10.

Рисунок 1.10. Пример дерева принятия решений.

Алгоритм деревьев описывает внутреннюю логику принятия решений, то есть его можно интерпретировать в отличии от алгоритмов искусственных нейронных сетей. Также время обучения меньше по сравнению с обучением нейронных сетей. Деревья решений могут обрабатывать большие объемы данных с хорошей точностью.

Мера выбора атрибута (ASM) — это правило для выбора критериев ветвления дерева, по которому ветвление происходит наилучшим образом.

Атрибут с наилучшими показателями будем выбран в качестве следующего по рангу атрибута. Если мы имеем дело с атрибутом, область определения которого имеет непрерывный характер, необходимо также определить точки разделения на ветви. Наиболее популярными метриками отбора являются коэффициент прироста информации, коэффициент усиления и индекс Джини[25].

Коэффициент прироста информации – это уменьшение энтропии системы, которой в нашем случае выступает дерево принятия решений. Коэффициент прироста информации представляет собой разницу между энтропией до разветвления на основе значения атрибутов и после:

$$Gain(A) = Info(D) - Info_A(D) \quad (4)$$

$$Info(D) = -\sum_{i=1}^{m} P_i log_2 P_i \ , \quad (5)$$

Где $Info(D)$ – энтропия системы по Шеннону [25];

P_i – вероятность нахождения системы в i-ом состоянии;

$$Info_A(D) = \sum_{j=1}^{V} \frac{|D_j|}{|D|} \cdot Info(D_j), \quad (6)$$

Где $\frac{|D_j|}{|D|}$ – вес j-ой системы, образованной после применения атрибута.

Разберем понятие коэффициента прироста информации на примере. В системе находится 9 синих шариков и 11 красных, в порядке как показано на рисунке 1.11.

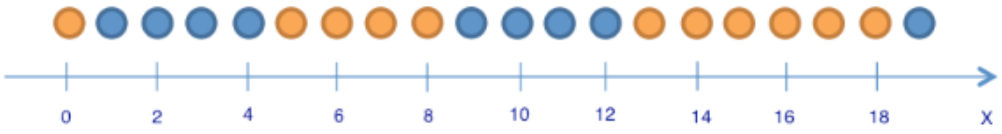

Рисунок 1.11

Энтропия системы по формуле 5:

$$Info(D) = -\frac{9}{20} log_2 \frac{9}{20} - \frac{11}{20} log_2 \frac{11}{20} \approx 1$$

Разобьем шарики на две группы, значение атрибута разбиение возьмем X = 12, тогда:

$$Info(D_1) = -\frac{8}{13} log_2 \frac{8}{13} - \frac{5}{13} log_2 \frac{5}{13} \approx 0{,}96$$

$$Info(D_2) = -\frac{1}{7} log_2 \frac{1}{7} - \frac{6}{7} log_2 \frac{6}{7} \approx 0{,}6$$

Вес первой подсистемы равен 13/20, второй – 7/20, таким образом можем рассчитать коэффициент прироста информации:

$$Gain(X = 12) = 1 - \frac{13}{20} \cdot 0.96 - \frac{7}{20} \cdot 0{,}6 \approx 0{,}16$$

Коэффициент прироста информации мало информативен при работе с атрибутами с большим количеством исходов. Поэтому в таких случаях используют коэффициент усиления. Коэффициент усиления – это коэффициент прироста информации, нормализованный с использованием разделенной информации (SplitInfo):

$$GainRation(A) = \frac{Gain(A)}{SplitInfo_A(D)} \quad (7)$$

$$SplitInfo_A(D) = -\sum_{j=1}^{V} \frac{|D_j|}{|D|} log_2 \frac{|D_j|}{|D|} \quad (8)$$

Из предыдущего примера:

$$SplitInfo_{X=12}(D) = -\frac{13}{20} log_2 \frac{13}{20} - -\frac{7}{20} log_2 \frac{7}{20} \approx 0{,}93$$

$$GainRation(X = 12) = \frac{0{,}16}{0{,}93} \approx 0{,}17$$

Для дискретных значений параметров также использует в качестве метрики метод Джини, при которой, подбирая значение атрибута, стремятся минимизировать следующую функцию:

$$\Delta Gini(A) = Gini(D) - Gini_A(D), \quad (9)$$

где

$$Gini(D) = 1 - \sum_{i=1}^{m} {P_i}^2 \quad (10)$$

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \quad (11)$$

Из предыдущего примера:

$$Gini(D) = 1 - \left(\frac{9}{20}\right)^2 - \left(\frac{11}{20}\right)^2 \approx 0{,}49$$

$$Gini(D_1) = 1 - \left(\frac{8}{13}\right)^2 - \left(\frac{5}{13}\right)^2 \approx 0{,}48$$

$$Gini(D_2) = 1 - \left(\frac{1}{7}\right)^2 - \left(\frac{6}{7}\right)^2 \approx 0{,}24$$

$$Gini_{X=12}(D) = \frac{13}{20} Gini(D_1) + \frac{7}{20} Gini(D_2) \approx 0{,}4$$

$$\Delta Gini(X = 12) = 0{,}49 - 0{,}4 = 0{,}09$$

Основная идея построения любого алгоритма дерева решений заключается в следующем[26]:

- Выбрать лучший атрибут с помощью мер выбора атрибутов (ASM), чтобы разделить записи;
- Сделать этот атрибут узлом принятия решений и разбить набор данных на еще более мелкие подмножества;
- Начинать построение дерева, повторяя этот процесс рекурсивно для каждого дочернего элемента до тех пор, пока одно из условий не будет выполнено:
    - Весь массив правил принадлежит к одному и тому же значению атрибута;
    - Создание более низкоуровневых атрибутов невозможно;
    - Больше нет данных для обучения.

Ниже описан пример обучения и применения дерева принятия решения на языке python 3.5, с использованием библиотек pandas и sklearn (Рисунок 1.12).

```
import pandas as pd
from sklearn.cross-validation import train_test_split
from sklearn.tree import DecisionTreeClassifier

dataset = pd.read_csv("Горобец_диплом.csv", header=None)
feature_cols = ['feature1', 'feature2','feature3']
x = dataset[feature_cols]
y = dataset.lable

x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, random_state=0)

model = DecisionTreeClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
```

Рисунок 1.12. Пример программы для создания и обучения дерева принятия решений.

Первый блок программы – подключение требуемых библиотек;

Второй блок – чтение датасета из csv файла;

Третий блок – разделение датасета на тренировочную и тестовую выборки в соотношении 3:1;

Четвертый блок – инициализация модели и ее обучение(расчет коэффициентов)(метод fit из библиотеки sklearn выполняет метод ASM);

Пятый блок – запуск обученной модели на тестовой выборке и запись результатов в массив y_pred.

Обычно при большом количестве классов и непрерывной области значений параметров одно дерево принятия решений показывает плохие результаты, по причине того, что обладает слабой обобщающей способностью. В таких случая используют архитектуру Случайный лес, которая представляет из себя несколько структур (ансамбль) деревьев принятия решений. Схематическое представление такой структуры представлено на рисунке 1.13.

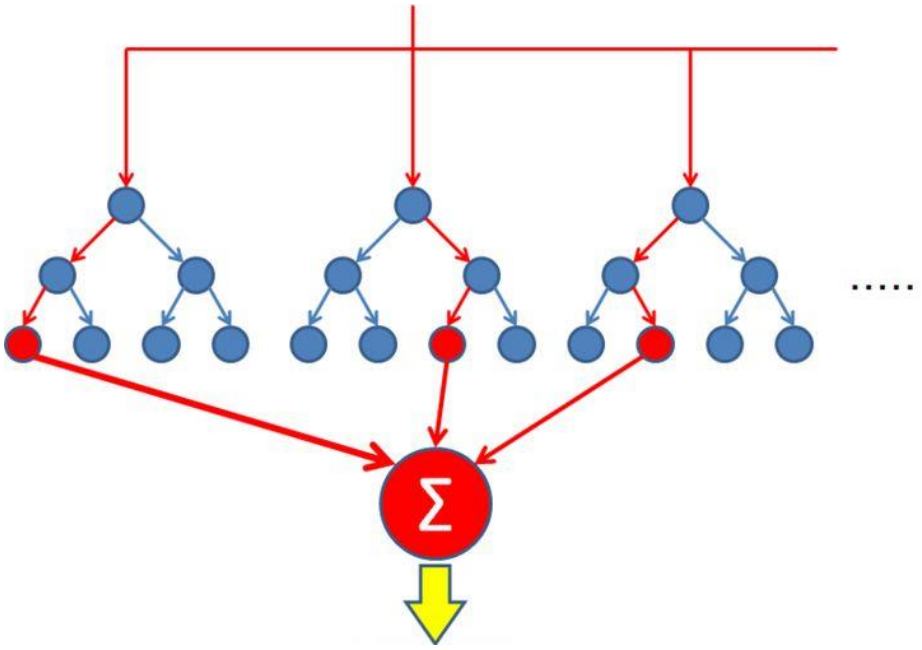

Рисунок 1.13. Схематическое представление архитектуры Случайный лес.

1.3.4. Искусственные нейронные сети.

Искусственные нейронные сети — это самая захватывающая и мощная отрасль машинного обучения. Это метод, который учит программы делать то, что естественно для человека: учиться на собственном примере.

В процессе обучения нейронных сетей компьютерная модель учится выполнять задачи классификации непосредственно по изображениям, тексту или звуку. Такие модели могут достигать самой современной точности, иногда превышающей производительность человеческого уровня.

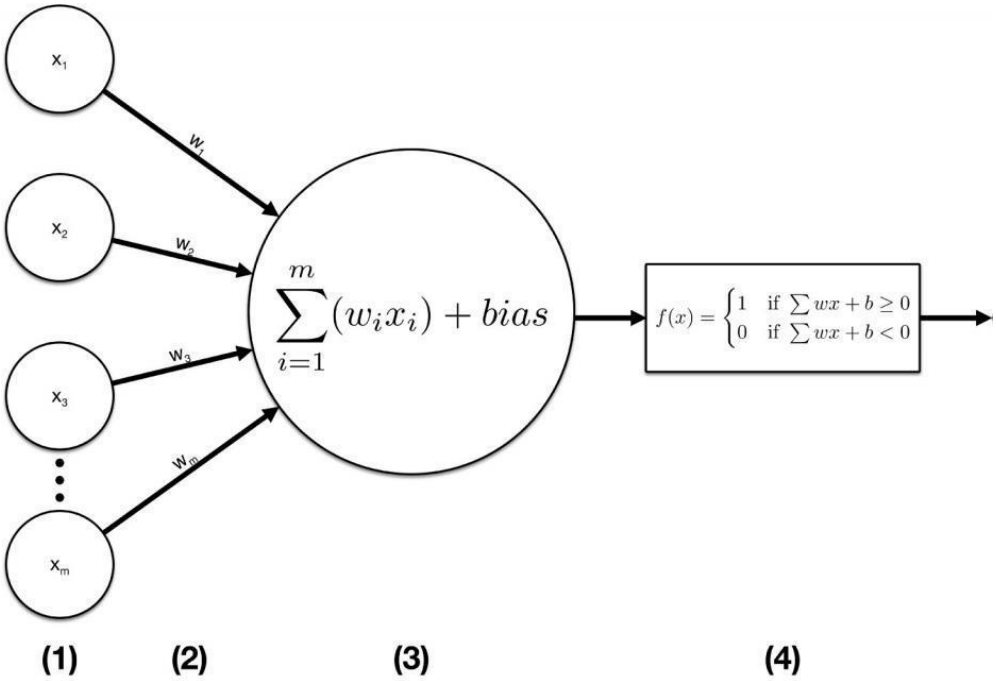

Рисунок 1.14. Модель персептрона.

На рисунке 1.14 изображена модель персептрона. Персептрон – это базовый элемент нейронной сети. Аналог одного нейрона в человеческом организме. Персептрон состоит из[27]:

1 – это входные параметры, каждый из параметров для нейронной сети представляется, как независимая переменная;

2 – каждый из входных параметров умножается на вес этого узла. По факту вес является подобием синапса в человеческом организме и указывает на прочность узла;

3 – далее полученные значения суммируются, к ним также прибавляется константа bias, которая отвечает за смещение функции активации вверх или вниз;

4 – далее к полученному значению применяется функция активации и устанавливаются правила, по которым можно интерпретировать результат.

Как мы видим структура персептрона очень похожа на структуру случайный лес. Это не случайно, поскольку и при том и при другом подходе в основе работы лежит теория графов.

Функция активации важна для искусственных нейронных сетей, чтобы они могли анализировать что-то действительно сложное. Их основное назначение-преобразование входного сигнала узла в нейросети в выходной сигнал. Этот выходной сигнал используется в качестве входного сигнала для следующего слоя в слоистой структуре всей нейронной сети.

Функция активации решает, должен ли нейрон быть активирован или нет, путем вычисления взвешенной суммы и дальнейшего добавления к ней смещения. Цель введения функции активации состоит в том, чтобы ввести нелинейность в выходной сигнал нейрона.

Если мы не применяем функцию активации, то выходной сигнал будет просто линейной функцией (полиномом первой степени). Полиномы первой степени ограничены по своей сложности, имеют меньшую мощность. Без функции активации модель не может изучать и моделировать сложные данные, такие как изображения, видео, аудио, речь и т. д.

Есть стандартный набор функций активации, такие как сигмоида, гиперболический тангенс и т.д. Также разработчик в праве написать собственную функцию активации для своей конкретной задачи.

Обучение в нейронной сети тесно связано с тем, как мы учимся в нашей обычной жизни и деятельности — мы выполняем действие и либо принимаем результат, либо корректируем себя учителем или тренером, чтобы понять, как лучше справиться с определенной задачей. На основе разницы между фактическим и прогнозируемым значением вычисляется значение ошибки, также называемое функцией затрат (Cost Function), и передается обратно через систему.

Функция затрат – половина квадратичной разницы между фактическим и выходным значением.

Для каждого слоя сети функция затрат анализируется и используется для корректировки порога и весов для следующего входного сигнала. Цель обучения -минимизировать функцию затрат. Чем ниже функция затрат, тем ближе фактическое значение к прогнозируемому. Таким образом, ошибка становится

незначительно меньше при каждом запуске, поскольку сеть учится анализировать значения[27].

Описанная выше процедура известна как обратное распространение и применяется непрерывно через сеть до тех пор, пока значение ошибки не будет сведено к минимуму.

Таким образом весь процесс обучения нейронной сети можно свести к двум операциям это:

1. Перемножение матрицы весов;
2. Метод приближенного вычисления – обратное распространение ошибки (градиентный спуск).

1.3.4.1. Полносвязная нейронная сеть.

Полносвязная нейронная сеть состоит из ряда полностью связанных слоев. Каждое выходное измерение зависит от каждого входного измерения[28]. В графическом виде полносвязный слой представлен на рисунке 1.14.

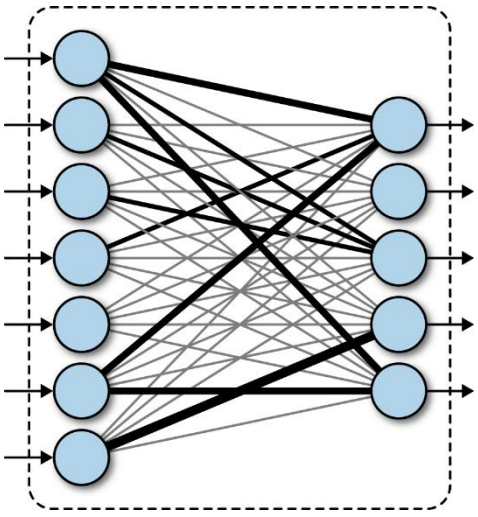

Рисунок 1.14. Графическое представление полносвязного слоя.

Полносвязная нейронная сеть – классическая модель искусственных нейронных сетей. Выходной сигнал такой нейронной сети можно посчитать по формуле:

$$y = f(\sum_{i=1}^{m} w_i x_i + bias) \quad (12)$$

1.3.4.2. Сверточная нейронная сеть.

Специфическим видом нейронной сети является сверточная сеть, которую обычно называют CNN. CNN, в частности, вдохновлены зрительной частью головного мозга. В коре головного мозга имеются небольшие участки клеток, чувствительные к определенным областям зрительного поля. Эта идея была расширена увлекательным экспериментом, проведенным Хьюбелом и Визелем в 1962 году[29]. В этом эксперименте исследователи показали, что некоторые отдельные нейроны в мозге активируются или срабатывают только при наличии ребер определенной ориентации, таких как вертикальные или горизонтальные ребра.

Сверточные нейронные сети являются одной из самых крупных инноваций в области компьютерного зрения. Они работают намного стабильнее, чем традиционное компьютерное зрение, и дают очень точные результаты. Эти нейронные сети доказали свою успешность во многих различных практических исследованиях и приложениях[30].

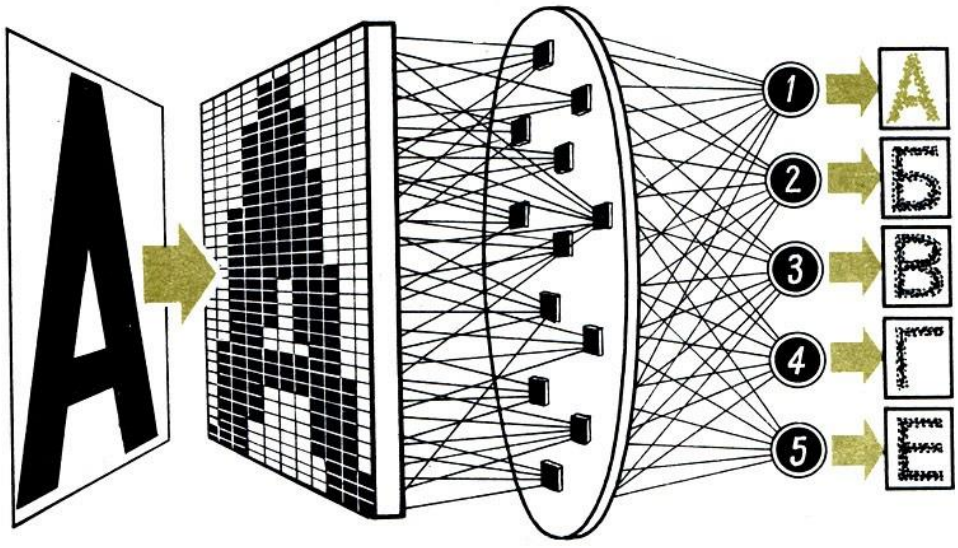

Рисунок 1.15. Наглядный пример работы сверточной нейронной сети.

Рисунок 1.15 показывает, что изображение передается в качестве входного сигнала в сеть, которая проходит через несколько сверточных слоев, которые по

факту понижают размерность исходного изображения, на основании данных последнего слоя, делается прогноз о том, что было показано на входе.

Функции, которые исполняет сверточная сеть[30]:

1) Сверточные слои: данный элемент организует операцию свертки, и по своей сути является матричным фильтром небольшого размера;
2) Слои субдискретизации: уменьшают размер изображения;
3) Полносвязный слой на выходе: используется для того, чтобы выровнять по весам высокоуровневые объекты, которые учатся анализировать сверточные слои. Основная цель использования – классификация по низкоуровневым фичам.

Как мы понимаем, сверточные нейронные сети можно использовать не только для анализа изображений, но также для анализа любой другой информации, представленной в виде массива.

1.3.4.3. Рекуррентная нейронная сеть.

Идея рекуррентных нейронных сетей заключается в использовании последовательной информации. В традиционной нейронной сети мы предполагаем, что все входы (и выходы) независимы друг от друга. Но для многих задач это очень плохой подход. Если вы хотите предсказать следующее слово в предложении, вам лучше знать, какие слова стояли перед ним. Такие сети называются рекуррентными, поскольку они выполняют одну и ту же задачу для каждого элемента последовательности, причем выход зависит от предыдущих вычислений. Теоретически Рекуррентные сети могут использовать информацию в произвольно длинных последовательностях, но на практике они ограничиваются лишь несколькими предыдущими шагами.

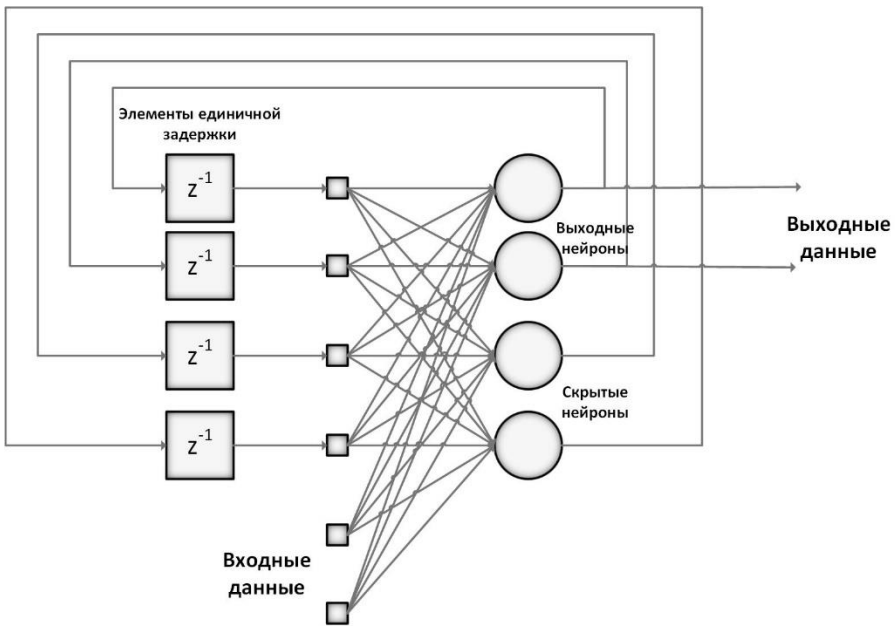

Рисунок 1.16. Структурная диаграмма рекуррентной нейронной сети.

На рисунке 1.16 можно наблюдать такой элемент как скрытые нейроны. Эти нейроны не имеют выходного слоя и служат своего рода памятью такой сети. По факту рекуррентная сеть выполняет одну и ту же задачу на каждом этапе, только с различными входными данными. Это значительно сокращает общее количество параметров, по сравнению с тем, если бы на входной слой подавался одновременно весь объем информации. Например, автоматический переводчик не переводит все предложение сразу, а по одному слову, при этом учитывая контекст[31].

1.3.5. Количественные метрики оценки качества работы моделей.

Для того, чтобы сравнить между собой две модели в контексте качества исполнения задачи используют метрики accuracy, precision, recall и F-мера.

Accuracy – как понятно из названия, это доля правильных ответов алгоритма[28]:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (13)$$

где TP – истинно положительные результаты;

TN – истинно отрицательные результаты;

FP – ложно положительные результаты (ошибки первого рода);

FN – ложно отрицательные результаты (ошибки второго рода).

Precision – доля объектов, названных классификатором положительными и при этом действительно являющимися положительными:

$$precision = \frac{TP}{TP+FP} \quad (14)$$

Recall – это отношение количества истинно положительных срабатываний на общее количество объектов этого класса:

$$recall = \frac{TP}{TP+FN} \quad (15)$$

Precision показывает способность отличать конкретный класс от других классов, recall – способность алгоритма в целом обнаруживать данный класс.

Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок[32].

Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. Самым распространенным является F-мера — среднее гармоническое precision и recall:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (16)$$

F-мера достигает единицы, при полноте и точности модели, и стремится к нулю, в случае когда модель не дообучена или тренировочный датасет не достаточно полный[33].

1.3.6. Выводы по разделу.

Были изучены современные подходы работы и анализа большого объема данных, а именно машинное обучение. Среди огромного разнообразия моделей машинного обучения мы рассмотрели наиболее часто используемые модели для решения задачи классификации, а именно регрессионные модели, деревья принятия решений и искусственные нейронные сети. Для первых двух были написаны примеры на языке python для инициализации модели, обучения ее и

дальнейшего вызова. Пример программы, построенной на нейронных сетях, будет показан далее.

В конце раздела мы рассмотрели метрики, по которым можно сравнить две различных модели. Эта информация нам понадобится в главе 3 «Экспериментальная часть» для выбора оптимальной модели.

1.4. Выводы по главе.

В данной главе была наработана теоретическая база для дальнейшей разработки устройства и проведения экспериментов с различными моделями машинного обучения.

В первом разделе мы рассмотрели возможности применения МЭМС акселерометров и гироскопов в приложениях для решения пользовательских задач в различных отраслях техники. Также изучили принципиальные схемы и принципы работы современных микро- вибрационных гироскопов и пьезорезестивных акселерометров. Важнейшей технологической операцией производства таких устройств является герметизация.

Во втором разделе рассмотрены современные беспроводные технологии передачи данных. Среди всего многообразия была выбрана технология Bluetooth, так она наиболее применима под нашу задачу. При выборе Bluetooth модуля мы будем в дальнейшем обращать внимание на такие характеристики, как класс устройства, версия спецификации (для связи с ПК нам необходим Bluetooth 2.1, далее мы поясним данное требование), и наличие в стеке протоколов RFCOMM.

В третьем разделе был рассмотрены методы анализа большого количества информации, а именно методы машинного обучения, как наиболее перспективные на сегодняшней день почти во всех областях науки и техники. Рассмотрены математические основы работы и правила обучения моделей машинного обучения. Выбраны метрики сравнения качества работы моделей между собой.

ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Hardware.

2.1.1. Проектирование устройства.

Перед тем как преступить к подбору компонентной базы и изготовлению устройства, необходимо ознакомиться с техническим заданием, выписка из которого представлена в Приложении А.

Выбор компонентной базы начинается с управляющего элемента (микроконтроллера или микропроцессора), в нашем случае необходимыми требованиями при выборе являются массогабаритные размеры, возможное количество подключенных устройств и скорость передачи данных. Наиболее популярными управляющими платами при прототипировании устройств являются платы линейки Arduino. Лимитирующим фактором здесь является габариты устройства. Поэтому, чтобы выполнить условия технического задания, мы выбрали Arduino Nano v3 с установленным 8-битным контроллером Atmega[34][35].

Следующим шагом является выбор датчиков для измерения скорости и угла поворота[36]. Современные микроакселерометры и микрогироскопы общего назначения являются пьезорезистивными и вибрационными соответственно, поэтому основными критериями при выборе этих элементов является: возможность работы при T = - 10 ^{o}C, возможность считывать показания с сенсоров минимум 30 раз в секунду, датчики должны являться трехосевыми, желательно, применение интегрированного решения. Одним из таких решений является продукт от Invensense микросхема MPU-6500[37].

MPU-6500 состоит из трёх независимых одноосных вибрационных датчиков угловой скорости, которые реагируют на вращение вокруг X-, Y-, Z-осей. Данный сигнал оцифровывается с помощью встроенного в плату 16-битного АЦП. Скорость оцифровки может программно варьироваться от 3.9 до 8000 выборок в секунду, а задаваемые пользователем фильтры низких частот

предоставляют широкий диапазон возможных частот среза. ФНЧ нужен, в том числе, чтобы убирать вибрации от моторов (как правило, выше 20-25 Гц).

Трехосевой акселерометр MPU-6500 использует для каждой оси отдельную пробную массу, которая смещается при возникновении ускорения вдоль данной оси (сигнал фиксируется емкостными датчиками). Архитектура MPU-6500 снижает подверженность температурному дрейфу и вариациям электропараметров. Масштабный коэффициент (scale factor — отношение изменения выходного сигнала к изменению входного измеряемого сигнала) калибруется на заводе и не зависит от напряжения питания. Каждый сенсор снабжен индивидуальным АЦП.

Третьим элементом устройства является устройство для передачи данных, Bluetooth модуль. Основным требованием здесь является возможность сопряжения как с ПК (в основном используется классический Bluetooth), так и со смартфоном (используется двухстековая схема BLE и классический Bluetooth). Ввиду этого ограничения мы не можем выбрать BLE устройство, из-за различий в стеке протоколов (BLE имеет в своем стеке протокол ATT, а классический Bluetooth – GATT). Также необходимым условием является возможность работы в режиме Transparent UART, то есть стек протоколов должен содержать протокол RFCOMM. Кроме того, должна обеспечиваться работа как в режиме master, так и в режиме slave. Одним из самых дешевых и простых в настройке таким устройством является HC-06[38].

На рисунке 2.1 представлена принципиальная схема устройства. На схеме цифрами отмечены: 1 – Arduino Nano, 2 – MPU-6500, 3 – микросхема LLC для конвертации уровня логического сигнала 5 В в 3,3 В и развязка для нее, 4 – Bluetooth модуль HC-06, 5 – элемент питания литий-полимерный аккумулятор на 7,4 В.

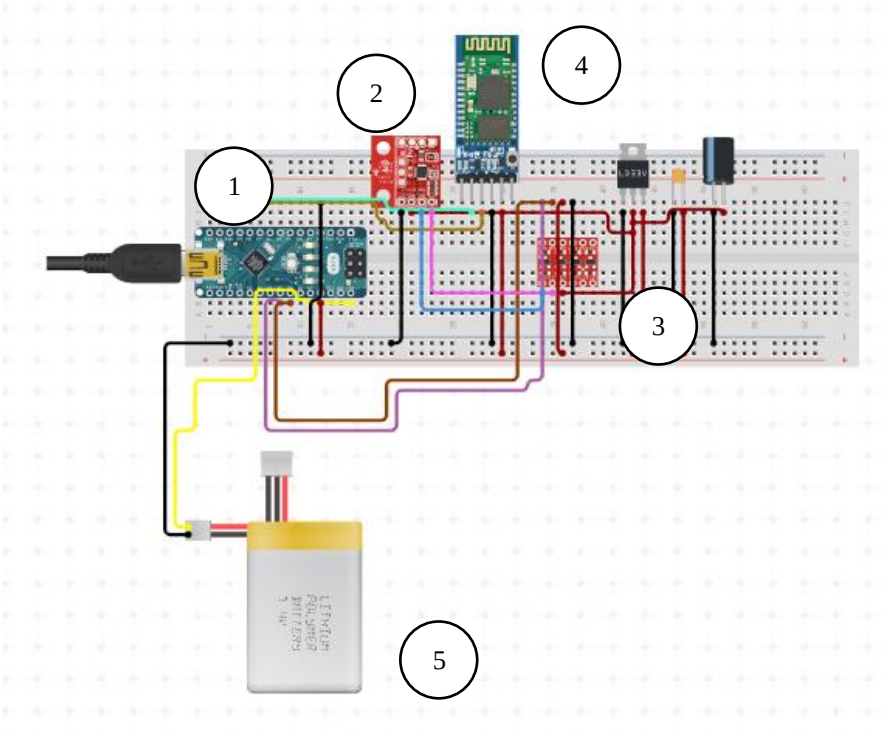

Рисунок 2.1. Принципиальная схема устройства.

Для удобного ношения на руке пользователя, данная схема была спаяна и помещена в термоусадку для фиксации элементов. Конечный вид прототипа изображен на рисунке 2.2.

Рисунок 2.2. Прототип устройства.

2.1.2. Настройка Bluetooth модуля.

Для обеспечения стабильной связи по каналу Bluetooth необходимо настроить определенным образом сам Bluetooth модуль. Для настройки Bluetooth модулей используются в соответствии со спецификацией Bluetooth AT, либо ASCI команды. HC-06 имеет протокол для настройки AT командами.

Для того, чтобы начать настройку, необходимо прошить Arduino кодом, который будет переводить Arduino в режим ретранслятора. Пример кода показан в приложении Б. Изначально HC-06 настроен на скорость передачи данных 9600 бод[39].

После прошивки для того, чтобы проверить установилось ли соединения, отправим в serial порт Arduino следующие команды: AT и AT+VERSION?. Если все было сделано правильно, в терминале должны появиться сообщения, как на рисунке 2.3[40].

Рисунок 2.3. Сообщения в терминале Arduino IDE в случае корректной работы устройства.

Необходимые команды:

1. AT+ORGL – сброс настроек;
2. AT+RMAAD – очистка реестра подключенных устройств;
3. AT+ROLE=1 – перевод устройств в режим master;
4. AT+BAUD115200 - установка скорости модуля на 115200 бод;
5. AT+UART=115200,0,0 – установка скорость UART 115200 бод;
6. AT+NAMEFL001 – переименование устройства, новое имя: FL001.

2.1.3. Прошивка устройства.

Код прошивки устройства написан на языке программирования C++, для загрузки скетча и подсветки синтаксиса используется компьютерная программа Arduino IDE рисунок2.4.

Рисунок 2.4. Окно Arduino IDE.

Скетч прошивки представлен в приложении В.

Два момента на которые необходимо обратить внимание:

1. В коде программы происходит инициализация интерфейса I2C для связи с MPU-6500;
2. Необходимо перевести выходной сигнал MPU-6500, который является сигналом от 16 разрядного АЦП в десятичные значения ускорения и угла поворота используя следующее выражение[37]:

$$A[i] = a[i]/16384 \quad (17)$$

2.1.4. Выводы по разделу

В этом разделе показано по какому принципу был произведен выбор компонентной базы и принципиальная схема подключения элементов. Также был показан алгоритм настройки Bluetooth модуля HC-06, а также итоговый код прошивки устройства.

2.2. Software

2.2.1. Инициализация последовательного порта.

Для того, чтобы передавать данные с устройства на сервер (ПК, смартфон), на стороне принимающего устройства необходимо инициализировать виртуальный последовательный COM порт. Только при инициализации COM порта можно использовать режим передачи данных Transparent UART. Этот режим работы эмулирует работу стандарта RS-232.

Операционная система Windows работает с COM портами, как с файлами, то есть может копировать из них информацию, а также посылать информацию в них. Как и UART RS-232, Transparent UART очень прост при разработке и для понимания принципов работы. Сообщения стандарта RS-232 формируются просто двоичной кодировкой и для увеличения скорости работы этот стандарт по умолчанию не предполагает проверку бита четности. Поэтому необходимо обеспечить максимальную сохранность сообщения, как со стороны передающего устройства, так и со стороны принимающего. Поэтому важнейшим условием при инициализации виртуального COM порта является синхронизация скоростей клиент-сервер[41].

Для инициализации COM порта был написан скетч на языке программирования Python 3.5 с использованием библиотеки serial. Данный скетч вы можете увидеть в приложении Г.

Как видно из кода из приложения, у виртуального COM порта была настроена такая же скорость, как у Bluetooth модуля устройства (115200 бод). А также в бесконечном цикле while происходит запись переданных данных с устройства в массив vals. То есть происходит то, о чем написано выше: программа берет данные с COM порта и копирует их в память компьютера с пометкой vals.

2.2.2. Алгоритмы постобработки.

Все методы классификации движений, которые существуют на сегодняшней день используют алгоритмы захвата движений, то есть устанавливают момент, когда началось движение. Такой подход достаточно точен, но, однако не обладает высокой скоростью быстродействия, поскольку анализ движения начинается после того, как сработает триггер, а также обычно триггер срабатывает, когда движение на 30-40 процентов выполнено. Также такой подход обладает высоким показателем ошибок второго рода, поскольку есть условие, что если сработал триггер, то движение обязательно должно быть предсказано, критерием выбора движения является наибольшая вероятность прогноза. А человек в это время мог просто жестикулировать.

В своей работе мы предлагаем подход классификации, при котором движение определяется по средневзвешенному последних 25 предсказаний модели машинного обучения. То есть на модель ежесекундно (даже, когда движение не выполняется) поступает 30-32 массива данных [ускорение по оси x, ускорение по оси y, ускорение по оси z, угол поворота по оси x, угол поворота по оси y, угол поворота по оси z], модель анализируя каждый такой массив предсказывает с какой вероятностью какое движение выполняется, то есть на выходе образуется массив, например такой [(движение1, 0,22), (движение2, 0,55), (движение3, 0,13), (движение4, 0,74)].

Таким образом на каждой итерации передачи данных с устройства происходит предсказание, которое записывается в общий массив предсказаний. На каждой итерации вероятности предсказаний складываются и делятся на 25 для того, чтобы минимизировать риск возникновения ошибки при похожих движениях. На каждой итерации получившиеся средневзвешенные значения сравниваются с эмпирически подобранными пороговыми значениями, и в тот момент, когда средневзвешенное превышает один из порогов, алгоритм сигнализирует, какое движение выполнялось и обнуляет массив с предсказаниями. Отрывок из основного кода, где реализован этот алгоритм представлен в приложение Д.

Также хотелось бы, чтобы классифицированные движения выполняли бы какую-нибудь полезную функцию. Например, имитировали нажатие клавиш на клавиатуре компьютера. Для решения этой задачи нами был написан скетч на языке программирования Python 3.5 с использованием библиотеки pynput.keyboard. Данный скетч представлен в приложении Е.

2.2.3. Выводы по разделу.

В данном разделе был показан скрипт для инициализации последовательного COM порта и показано, почему нам это необходимо.

Был показан алгоритм постобработки, который принципиально отличается от традиционных методов захвата движений. Он основан на расчете средневзвешенного от предсказаний модели машинного обучения. В ходе сравнения характеристик (здесь мы их не стали демонстрировать, но вы можете с ними ознакомиться в данной публикации[42]), мы установили, что такой подход в 2-2,5 раза быстрее, традиционных методов, а также сопровождается меньшим количеством ошибок второго рода, однако стоит отметить, что некоторые движения могут быть не распознаны.

Также в заключение был продемонстрирован скрипт для имитации нажатия клавиш на клавиатуре компьютера.

2.3. Выводы по главе.

В данной главы были рассмотрены основные моменты проектирования устройства, написания прошивки для него, а также написания программы для принимающего устройства. Была продемонстрирована принципиальная схема устройства, внешний вид готового устройства и условия, опираясь на которые, была выбрана элементная база.

Во втором разделе был продемонстрированы алгоритмы постобработки, а также причины, по которым мы отказались от традиционных методов захвата движений.

В следующей главе мы обсудим способ набора датасета для обучения моделей машинного обучения, проведем эксперименты с различными моделями, сравним их между собой на основании метрик, предложенных в первой главе, а также продемонстрируем итоговый вариант алгоритма для классификации движений.

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

3.1. Сбор датасета.

Для обучения и тестирования моделей нам необходимо набрать датасет. Для его набора был написан скрипт, который показывает пользователю, какое движение необходимо выполнить, записывает показания датчиков, которые были отправлены по Bluetooth каналу и подписывает их соответствующим типом движения.

Данный скрипт представлен в приложении Ж.

Для проведения экспериментов мы использовали 8 типов жестов:

1. Взмах рукой вверх;
2. Взмах рукой вниз;
3. Взмах рукой вправо;
4. Взмах рукой влево;
5. Отведение руки от себя;
6. Отведение руки к себе;
7. Поворот предплечья по часовой стрелке;
8. Поворот предплечья против часовой стрелки;

В сборе датасета приняли участие четыре человека, каждый из которых записал ровно 400 раз каждое движение. Таким образом общий объем датасета составил 12 800 образцов, 1600 образцов на каждый тип движения. Датасет представляет собой два csv файла – первый с данными, полученными от устройства, второй соответствующий тип движения. Пример датасета представлен на рисунках 3.1 и 3.2.

Весь датасет был разделен на 80% обучающей выборки и 20% тестовой.

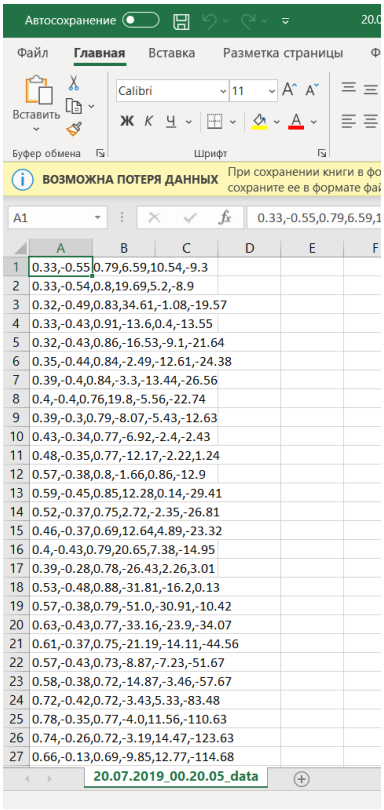

Рисунок 3.1. Файл с показаниями устройства.

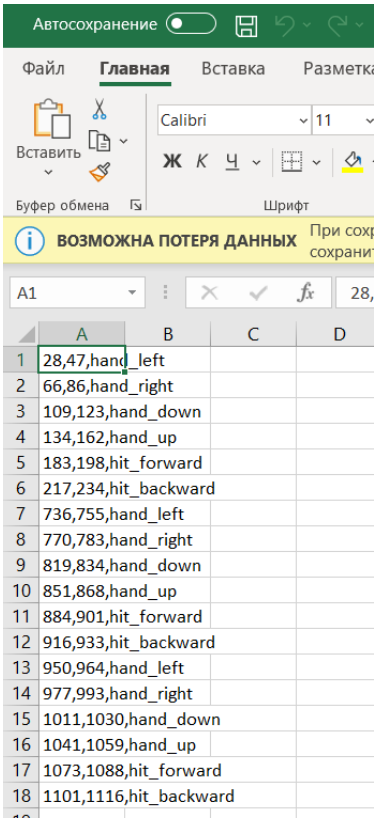

Рисунок 3.2. Файл с подписями.

3.2. Архитектура решающих деревьев.

Первой тестируемой архитектурой были выбраны решающие деревья. Время обучения на видеокарте NVIDIA GeForceMX 130 составило 8 минут 34 секунды.

Матрица ошибок представлена в таблице 1.

Таблица 1

	Реальные данные								
Предсказанные данные		Вверх	Вниз	Вправо	Влево	Вперед	Назад	По часовой	Против часовой
	Вверх	193	71	62	52	48	31	6	13
	Вниз	0	236	24	18	0	0	0	0
	Вправо	12	1	142	1	0	3	0	0
	Влево	23	11	0	158	2	0	0	0
	Вперед	11	0	27	42	257	18	0	0
	Назад	2	1	6	11	0	244	0	0
	По часовой	71	0	52	6	1	24	314	0
	Против Часовой	8	0	7	33	12	0	0	307

Метрики классификатора, построенного на данной архитектуре представлены на рисунке 3.3 и в таблице 2.

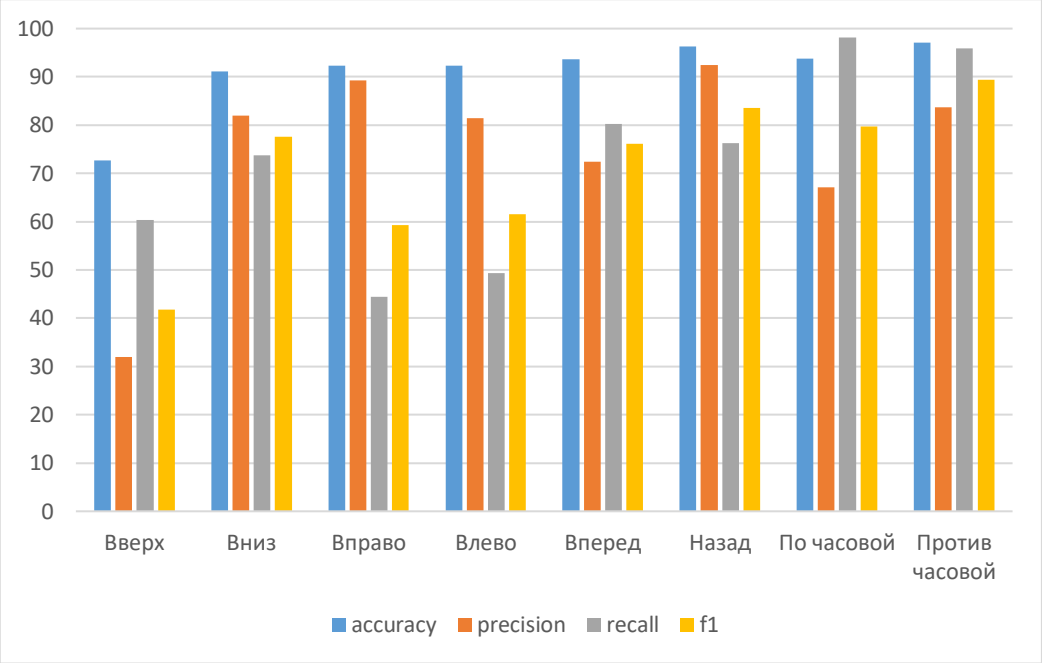

Рисунок 3.3. Метрики классификатора «Решающие деревья»

Таблица 2

	accuracy	precision	Recall	f1
Вверх	72,7	32,01	60,31	41,82
Вниз	91,17	81,94	73,75	77,63
Вправо	92,38	89,31	44,38	59,3
Влево	92,27	81,44	49,38	61,48
Вперед	93,71	72,39	80,31	76,14
Назад	96,25	92,42	76,25	83,56
По часовой	93,75	67,09	98,13	79,69
Против часовой	97,15	83,65	95,94	89,37

3.3. Полносвязная нейронная сеть.

Второй исследуемой моделью является полносвязная нейронная сеть, составленная из 6 слоев. Количество нейронов в каждом слое:

1. 310;
2. 110;
3. 110;
4. 110;
5. 115;
6. 8.

Функция активации выходного слоя – SoftMax. Модель обучалась в 7 эпох, время обучения на видеокарте NVIDIA GeForceMX 130 составило 18 минут 13 секунд.

Матрица ошибок представлена в таблице 3.

Таблица 3

	Реальные данные								
Предсказанные данные		Вверх	Вниз	Вправо	Влево	Вперед	Назад	По часовой	Против часовой
	Вверх	187	15	15	17	12	11	5	8
	Вниз	15	241	11	14	8	7	1	0
	Вправо	22	8	223	9	7	9	2	1
	Влево	25	13	9	219	9	5	1	2
	Вперед	19	10	19	16	274	14	2	2
	Назад	12	12	15	13	1	252	1	1
	По часовой	24	11	16	11	1	19	307	1
	Против Часовой	16	10	12	21	8	3	1	305

Метрики классификатора, построенного на данной архитектуре представлены на рисунке 3.4 и в таблице 4.

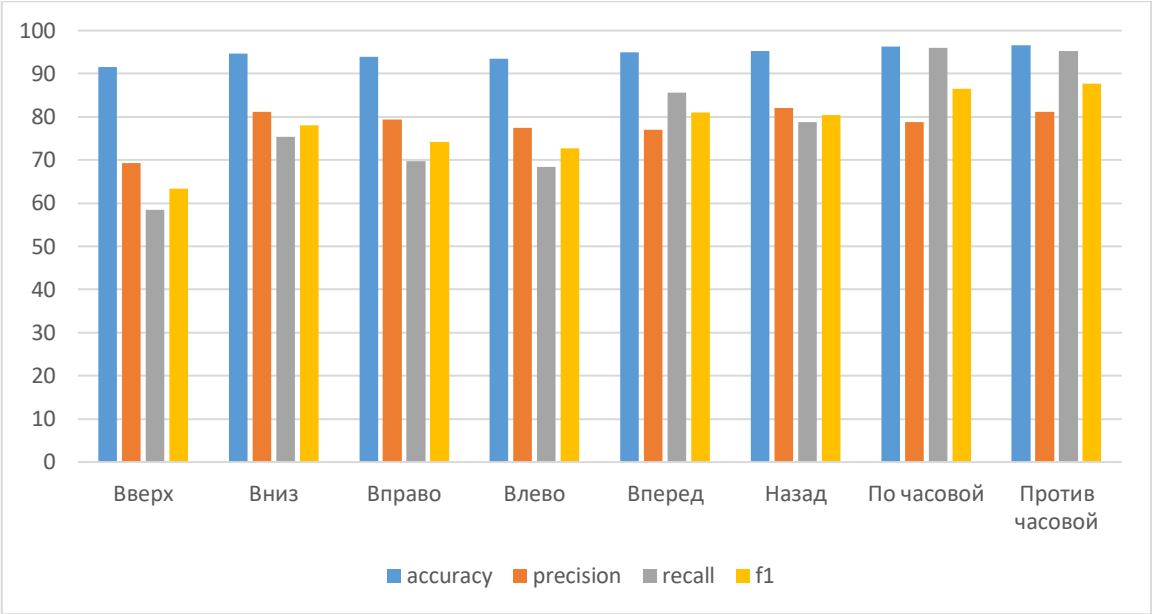

Рисунок 3.4. Метрики классификатора «Полносвязная нейронная сеть»

Таблица 4

	accuracy	precision	recall	f1
Вверх	91,56	69,26	58,44	63,39
Вниз	94,73	81,14	75,31	78,12
Вправо	93,95	79,36	69,69	74,21
Влево	93,55	77,39	68,44	72,64
Вперед	95	76,97	85,63	81,07
Назад	95,2	82,08	78,75	80,38
По часовой	96,25	78,72	95,94	86,48
Против часовой	96,64	81,12	95,31	87,64

3.4. Сверточная нейронная сеть.

Третьей исследуемой моделью является сверточная нейронная сеть, составленная из 5 слоев, первые 3 из которых сверточных, и два полносвязных. Количество нейронов в каждом слое:

Функция активации выходного слоя – SoftMax. Модель обучалась в 10 эпох, время обучения на видеокарте NVIDIA GeForceMX 130 составило 12 минут 44 секунд.

Матрица ошибок представлена в таблице 5.

Таблица 5

	Реальные данные								
Предсказанные данные		Вверх	Вниз	Вправо	Влево	Вперед	Назад	По часовой	Против часовой
	Вверх	234	11	4	13	12	7	6	9
	Вниз	9	283	11	3	1	0	0	0
	Вправо	7	1	287	1	3	2	1	0
	Влево	21	9	0	269	3	1	0	2
	Вперед	11	3	3	8	299	0	0	0
	Назад	11	4	3	3	0	291	0	0
	По часовой	18	5	8	6	0	10	313	0
	Против Часовой	9	4	4	17	2	9	0	309

Метрики классификатора, построенного на данной архитектуре представлены на рисунке 3.5 и в таблице 6.

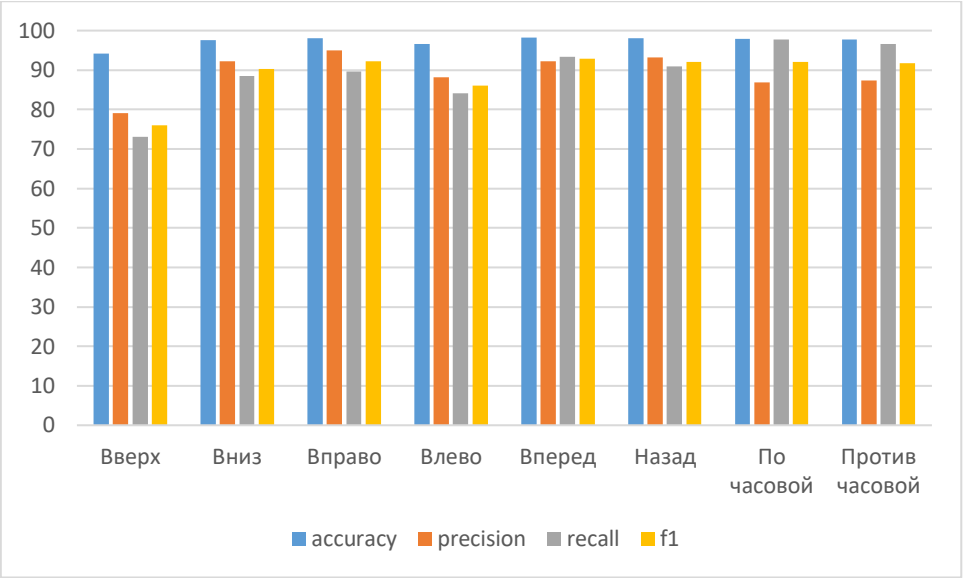

Рисунок 3.5. Метрики классификатора «Сверточная нейронная сеть»

Таблица 6

	accuracy	precision	recall	f1
Вверх	94,22	79,05	73,13	75,97
Вниз	97,62	92,18	88,44	90,27
Вправо	98,13	95,03	89,69	92,28
Влево	96,6	88,2	84,06	86,08
Вперед	98,2	92,28	93,44	92,86
Назад	98,05	93,27	90,94	92,09
По часовой	97,89	86,94	97,81	92,06
Против часовой	97,81	87,29	96,56	91,69

3.5. Выводы по главе.

В главе были описаны результаты экспериментов с различными моделями машинного обучения, а также был представлен метод сбора датасета. Обучающая выборка составляла 10 240 образцов движений, тестовая выборка – 2 560. Для сравнения моделей были выбраны следующие метрики: accuracy, precision, recall, f1.

Наилучшими характеристиками обладает модель с тремя сверточными слоями и двумя полносвязными. Поэтому в конечном варианте прототипа мы будем использовать эту модель. Ее архитектура представлена в формате json файла в приложении З. Итоговый вариант программы, которая должна быть запущена на принимающем и обрабатывающем устройстве представлена в приложении И. Эта программа написана на языке программирования python 3.4, с использованием библиотек keras, numpy, sklearn, tensorflow.

ЗАКЛЮЧЕНИЕ

В ходе исследования были решены следующие задачи:

1. Проведен обзор литературы на тему современных технологий изготовления IoT датчиков. Выяснено, что на сегодняшний день наиболее популярными в пользовательском сегменте пьезорезистивные акселерометры и вибрационные гироскопы;
2. Проведен обзор литературы на тему современных технологий передачи данных. Все технологии беспроводной передачи данных можно разделить на четыре класса: персональные сети, локальные, сети в масштабах города и глобальные. Исследована спецификация Bluetooth, стек протоколов этой технологии и сферы ее применения;
3. Изучены основы методов анализа больших объемов данных, а именно современные подходы машинного обучения;
4. Написано техническое задание на изготовление прототипа, выбрана компонентная база и как следствие опытный образец был изготовлен;
5. Был разработан алгоритм настройки Bluetooth модуля и прошивка для устройства;
6. Был разработан алгоритм для замены традиционного алгоритма захвата движений, основанный на вычислении средневзвешенного предсказания модели машинного обучения;
7. Проведены эксперименты с тремя различными моделями машинного обучения: решающие деревья, полносвязная нейронная сеть, сверточная нейронная сеть. В ходе испытаний наилучшие характеристики показала сверточная нейронная сеть, в результате чего она и была использована в конечном варианте программы для принимающего и обрабатывающего устройства.

Цель была достигнута, гипотеза о возможности создания простого гаджета для распознавания движений рук нашла свое подтверждение.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Gregorio F. et al. Internet of Things // Signals and Communication Technology. 2020.
2. Iannacci J. Introduction to MEMS and RF-MEMS: From the early days of microsystems to modern RF-MEMS passives // RF-MEMS Technology for High-Performance Passives. 2017. 1–39 p.
3. Song C., Shinn M. Commercial vision of silicon-based inertial sensors // Sensors Actuators, A Phys. 1998.
4. Sherman S.J. et al. A low cost monolithic accelerometer; Product/technology update // Technical Digest - International Electron Devices Meeting, IEDM. 1992.
5. Iannacci J. Reliability of MEMS: A perspective on failure mechanisms, improvement solutions and best practices at development level // Displays. 2015.
6. Grace R.H. Growing presence of MEMS and MST in automotive applications // Sensors (Peterborough, NH). 1999.
7. Tanenhaus M. et al. Precision navigation for UAVs, mini-munitions, and handhelds through application of low cost accurate MEMS IMU/INS technology // Record - IEEE PLANS, Position Location and Navigation Symposium. 2008.
8. Chan H.C.Y. Internet of Things Business Models // J. Serv. Sci. Manag. 2015.
9. Petersen K.E. Silicon as a Mechanical Material // Proc. IEEE. 1982.
10. Bhat K.N. Micromachining for microelectromechanical systems // Def. Sci. J. 1998. Vol. 48, № 1. P. 5–19.
11. Ergun A.S. et al. MEMS/NEMS Techniques and Applications // MEMS/NEMS. 2006.
12. Lainé J., Mougenot D. A high-sensitivity MEMS-based accelerometer // Lead. Edge. 2014.
13. Shkel C.A. and A. MEMS vibratory gyroscopes: structural approaches to improve robustness // Springer Science & Business Media. 2013.
14. Apostolyuk V. Theory and Design of Micromechanical Vibratory Gyroscopes // MEMS/NEMS. 2007.

15. Ur-Rehman O., Zivic N. Wireless communications // Signals and Communication Technology. 2018.
16. McDermott-Wells P. What is Bluetooth? // IEEE Potentials. 2004.
17. Camps-Mur D., Garcia-Saavedra A., Serrano P. Device-to-device communications with WiFi direct: Overview and experimentation // IEEE Wirel. Commun. 2013.
18. Maier M. et al. WiMAX // FiWi Access Networks. 2012.
19. Dahlman E., Parkvall S., Skold J. 4G: LTE/LTE-Advanced for Mobile Broadband // 4G: LTE/LTE-Advanced for Mobile Broadband. 2013.
20. Kaur J., Kaur R., Kaur M. Bluetooth Technology // Int. J. Eng. Comput. Sci. 2016.
21. Bluetooth S. Specification of the Bluetooth system // Core, version. 2005.
22. Bluetooth Special Interest Group (SIG). Bluetooth Core Specification Version 5.0 // Bluetooth Core Specif. Version 4.2. 2016. № December. P. 2684.
23. Amari S. ichi. Machine Learning // Applied Mathematical Sciences (Switzerland). 2016.
24. Carè A., Camporeale E. Regression // Machine Learning Techniques for Space Weather. 2018.
25. Quinlan J.R. Induction of decision trees // Mach. Learn. 1986.
26. Podgorelec V., Zorman M. Decision Tree Learning // Encyclopedia of Complexity and Systems Science. 2015.
27. Khan G.M. Artificial neural network (ANNs) // Studies in Computational Intelligence. 2018.
28. Cook T.R. Neural Networks // Advanced Studies in Theoretical and Applied Econometrics. 2020.
29. Hubel D.H., Wiesel T.N. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex // J. Physiol. 1962.
30. Zhang Q. et al. Recent advances in convolutional neural network acceleration // Neurocomputing. 2019.
31. Caterini A.L., Chang D.E. Recurrent neural networks // SpringerBriefs in

Computer Science. 2018.

32. Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка // Computer. 2019.
33. Tu Y. Machine learning // EEG Signal Processing and Feature Extraction. 2019.
34. www.mouser.com. Arduino Nano (V3.0) user manual. 2009. P. 5–7.
35. Performance H. et al. Features – Optional Boot Code Section with Independent Lock Bits – Programming Lock for Software Security I / O and Packages Operating Voltage : Temperature Range : Speed Grade : Microcontroller with 8K Bytes Programmable Flash ATmega48 / V ATmega88 / V A.
36. Xu R., Zhou S., Li W.J. MEMS accelerometer based nonspecific-user hand gesture recognition // IEEE Sens. J. 2012.
37. Invensence. MPU-9250 Product Specification Revision 1.1 MPU-9250 Product Specification [Electronic resource] // Invensence. 2016.
38. Feasycom S., Co T. HC-06 Document Type : Document Version : Release Date : HC-06 HC-06 Datasheet Version Number. 2017. P. 1–17.
39. Electrónica 60 Norte. HC05 Bluetooth Module // Data Sheet. 2016.
40. Sayem. AT Command Mode of HC-05 and HC-06 Bluetooth Module [Electronic resource] // Autodesk. 2016.
41. Axelson J. Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems // Lakeview Research. 2007.
42. Tung H.Y.F. et al. Self-supervised learning of motion capture // Advances in Neural Information Processing Systems. 2017.

ПРИЛОЖЕНИЕ А

Выписка из технического задания

Назначение и цели создания устройства

Назначение устройства

Устройство FL001 представляет собой прототип устройства для захвата движений руки и трансляции этих данных по беспроводному каналу передачи данных на компьютер и смартфон. Предназначено для проверки гипотезы о возможности классификации движений.

Цели создания устройства

Планируется использование FL001 в следующих целях:

- Набор датасета;
- Проведение экспериментов и испытаний;
- Демонстрация концепции, предлагаемой проектом Gesture.

Требования к устройству

Основными применяемыми при создании и эксплуатации устройства принципами должны являться:

- Обеспечение связи и с ПК, и со смартфонами;
- Передача не менее 30 координат в секунду;
- Устройство необходимо надевать на руку пользователя.

Требования к устройству в целом

Устройство должно считывать и передавать данные об ускорении и угле поворота в трехмерном пространстве, в диапазоне температур выше 0 ^{o}C с периодом минимум раз в 35 мс.

Требования к габаритным размерам устройства:

Размеры устройства не более: 50*20*30 мм;

Масса не более: 50 г;

Требования к надежности

Особых требований к надежности на этапе прототипирования нет.

Требования к диапазону температур работы устройства

Устройство должно стабильно работать в диапазоне температур: T = -10 … +40 ºC.

Требования к основным электрическим параметрам устройства при T= +25 ºC

При нормальной температуре окружающей среды **Устройство** должна иметь следующие электрические параметры:

Рабочие напряжение: 5 В;

Входное напряжение, напряжение питания: 7-12 В;

Ток питания: 0,5-1 А*ч;

Тактовая частота: 16 МГц;

Скорость передачи данных: 115200 бод.

Порядок контроля и приемки устройства

В процессе приемки устройства в эксплуатацию выполняются проверки на соответствия настоящих требований, функционала, исполняющегося устройством, а также на наличие дефектов.

ПРИЛОЖЕНИЕ Б

Код для перевода Arduino в режим ретрансляции

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3);

void setup()  {
  pinMode(2,INPUT);
  pinMode(3,OUTPUT);
  Serial.begin(9600);
  mySerial.begin(9600);
  Serial.println("start prg");
}

void loop() {
  if (mySerial.available()) {
    char c = mySerial.read();
    Serial.print(c);
  }
  if (Serial.available()) {
    char c = Serial.read();
    mySerial.write(c);
  }
}
```

ПРИЛОЖЕНИЕ В

Код прошивки Arduino.

```
#include <Wire.h>
#include <I2Cdev.h>
#include <MPU9250.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // указываем пины rx и tx соответственно
MPU9250 accelgyro;
I2Cdev   I2C_M;
uint8_t buffer_m[6];
int16_t ax, ay, az;
int16_t gx, gy, gz;
float heading;
float tiltheading;
double Axyz[3];
double Gxyz[3];

void setup()
{
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
  //подключаемся к шине I2C (I2Cdev не может сделать это самостоятельно)
  Wire.begin();
  // инициализация подключения в Мониторе порта
  Serial.begin(115200);
  mySerial.begin(115200);
  Serial.println("start prg");
  // Инициализация устройства
  Serial.println("Initializing I2C devices...");
  accelgyro.initialize();
  // Подтверждение подключения
  Serial.println("Testing device connections...");
  Serial.println(accelgyro.testConnection()    ?    "MPU9250    connection
successful" : "MPU9250 connection failed");
  Serial.println("     ");
}
void loop()
{

  getAccel_Data();             // Получение значений Акселерометра
```

```
        getGyro_Data();               // Получение значений Гироскопа
        getHeading();                   // после чего мы получаем откалиброванные
значения углов поворота
        getTiltHeading();             // и наклона

        mySerial.print(Axyz[0]);
        mySerial.print(" ");
        mySerial.print(Axyz[1]);
        mySerial.print(" ");
        mySerial.print(Axyz[2]);
        mySerial.print(" ");

        mySerial.print(Gxyz[0]);
        mySerial.print(" ");
        mySerial.print(Gxyz[1]);
        mySerial.print(" ");
        mySerial.println(Gxyz[2]);
      }
      void getAccel_Data(void)
      {
        accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz);
        Axyz[0] = (double) ax / (double) 16384;
        Axyz[1] = (double) ay / (double) 16384;
        Axyz[2] = (double) az / (double) 16384;
      }
      void getGyro_Data(void)
      {
        accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz);
        Gxyz[0] = (double) gx / (double) 16384;
        Gxyz[1] = (double) gy / (double) 16384;
        Gxyz[2] = (double) gz  / (double) 16384;
      }
```

ПРИЛОЖЕНИЕ Г

Код для инициализации виртуального COM порта.

```
import serial
import time

port = "COM9"
speed = 115200
ser = serial.Serial(port, speed)

vals = []
counter = 0
while True:
      line = ser.readline().decode().strip()
      values = line.split()
      print(values)

      if len(values) == 6:
            values = [float(i) for i in values]
            vals.append(values)
```

ПРИЛОЖЕНИЕ Д.

Алгоритм расчета средневзвешенного предсказания.

```
last_predictions = []
threshes = [0.5, 0.5, 0.5, 0.95, 0.8, 0.65]
last_predictions_size = 25
last_cd_range = 14
last_counter = None

while True:
    line = ser.readline().decode().strip()
    values = line.split()

    if len(values) == 6:
        values = [float(i) for i in values]
        vals.append(values)

        last = np.array([vals[-30:]])
        with graph.as_default():
            if last_counter is None:
                current_prediction = loaded_model.predict(last)      # вероятности классов
                last_predictions += current_prediction.tolist()
                last_predictions = last_predictions[-last_predictions_size:]

                last_probs = [0 for i in range(6)]
                for j in range(6):
                    for i in range(len(last_predictions)):
                        last_probs[j] += last_predictions[i][j] / last_predictions_size

                # если трешхолд больше, чем вероятность, то жест не предсказан
                difs = [threshes[i] - last_probs[i] for i in range(6)]
                min_dif = min(difs)
                if min_dif <= 0:
                    current_gesture = difs.index(min_dif)
                    label = encoder.inverse_transform([current_gesture])[0]
                    print(label)
                    keyboard.press(gestures_to_keys[label])
```

```
            last_counter = 0
    elif last_counter is not None and last_counter < last_cd_range:
        last_counter += 1
        last_predictions = []
    elif last_counter is not None and last_counter >= last_cd_range:
        last_counter = None
        last_predictions = []
```

ПРИЛОЖЕНИЕ Е

Скетч для имитации нажатия клавиш

```
from pynput.keyboard import Key, Controller
keyboard = Controller()
gestures_to_keys = {
    "hand_left": Key.left,
    "hand_right": Key.right,
    "hand_up": Key.up,
    "hand_down": Key.down,
    "hit_forward": Key.esc,
    "hit_backward": Key.f5
}
keyboard.press(gestures_to_keys[label])
```

ПРИЛОЖЕНИЕ Ж

Скрипт для сбора датасета.

```
import csv
from time import gmtime, strftime

import msvcrt
import serial

port = "COM9"
speed = 115200
ser = serial.Serial(port, speed)

available_gestures = [
    "hand_left",
    "hand_right",
    "hit_forward",
    "hit_backward",
    "hand_down",
    "hand_up",
]
current_gesture_pointer = 0

cur_time = strftime("%d.%m.%Y_%H.%M.%S", gmtime())
data = []
indices = []

cur_circle = [0 for i in range(6)]  # start point of moving

t1 = None
counter = 1
index = 0


def out_of_sphere(vals, threshold=1):
    global cur_circle

    distance = sum([val ** 2 for val in vals[:3]])
    return distance > threshold ** 2


def out_of_segment(vals, threshold=1):
```

```
        global cur_circle

        dists = [abs(vals[i] - cur_circle[i]) for i in range(3)]
        is_bigger = [dist > threshold for dist in dists]

        # any value is greater than threshold
        return any(is_bigger)


    def save_file():
        global cur_time, data, indices

        with open(cur_time + "_data.csv", "a+", newline='') as f:
            writer = csv.writer(f)
            for values in data:
                writer.writerow(values)

        with open(cur_time + "_indices.csv", "a+", newline='') as f:
            writer = csv.writer(f)
            for values in indices:
                writer.writerow(values)

        data = []
        indices = []


    print("Do                                                          gesture:
{}".format(available_gestures[current_gesture_pointer]))
    try:
        while True:
            line = ser.readline().decode().strip()
            values = line.split()

            if len(values) == 6:
                values = [float(i) for i in values]
                data.append(values)

                if not t1 and msvcrt.kbhit():
                    t1 = index
                    print("Record start")
                    msvcrt.getch()
                elif t1 and msvcrt.kbhit():
                    t2 = index
```

```
                        indices.append((t1,                                    t2,
available_gestures[current_gesture_pointer]))
                        save_file()
                        print("Record end")

                        current_gesture_pointer += 1
                        if current_gesture_pointer >= len(available_gestures):
                            current_gesture_pointer = 0
                        msvcrt.getch()
                        t1 = None

                        print(counter)
                        counter += 1
                        print()
                        print("Do                                          gesture:
{}".format(available_gestures[current_gesture_pointer]))

                    index += 1


        except KeyboardInterrupt:
            print("ending session")
            save_file()
            print("files saved")
```

ПРИЛОЖЕНИЕ 3

Архитектура итоговой модели. Сверточная нейронная сеть.


```
{
  "class_name": "Sequential",
  "config": {
    "layers": [
      {
        "class_name": "Conv1D",
        "config": {
          "bias_regularizer": null,
          "data_format": "channels_last",
          "name": "conv1d_4",
          "strides": [
            1
          ],
          "bias_constraint": null,
          "bias_initializer": {
            "class_name": "Zeros",
            "config": {}
          },
          "filters": 8,
          "kernel_constraint": null,
          "activation": "relu",
          "trainable": true,
          "kernel_size": [
            5
          ],
          "padding": "same",
          "activity_regularizer": null,
          "kernel_initializer": {
            "class_name": "RandomNormal",
            "config": {
              "seed": null,
              "mean": 0.0,
              "stddev": 0.05
            }
          },
          "use_bias": true,
          "dilation_rate": [
            1
          ],
```

```
      "kernel_regularizer": null
    }
  },
  {
    "class_name": "MaxPooling1D",
    "config": {
      "pool_size": [
        2
      ],
      "data_format": "channels_last",
      "trainable": true,
      "padding": "valid",
      "name": "max_pooling1d_4",
      "strides": [
        2
      ]
    }
  },
  {
    "class_name": "Conv1D",
    "config": {
      "bias_regularizer": null,
      "data_format": "channels_last",
      "name": "conv1d_5",
      "strides": [
        1
      ],
      "bias_constraint": null,
      "bias_initializer": {
        "class_name": "Zeros",
        "config": {}
      },
      "filters": 10,
      "kernel_constraint": null,
      "activation": "relu",
      "trainable": true,
      "kernel_size": [
        5
      ],
      "padding": "valid",
      "activity_regularizer": null,
      "kernel_initializer": {
        "class_name": "RandomNormal",
```

```
      "config": {
        "seed": null,
        "mean": 0.0,
        "stddev": 0.05
      }
    },
    "use_bias": true,
    "dilation_rate": [
      1
    ],
    "kernel_regularizer": null
  }
},
{
  "class_name": "MaxPooling1D",
  "config": {
    "pool_size": [
      2
    ],
    "data_format": "channels_last",
    "trainable": true,
    "padding": "valid",
    "name": "max_pooling1d_5",
    "strides": [
      2
    ]
  }
},
{
  "class_name": "Conv1D",
  "config": {
    "bias_regularizer": null,
    "data_format": "channels_last",
    "name": "conv1d_6",
    "strides": [
      1
    ],
    "bias_constraint": null,
    "bias_initializer": {
      "class_name": "Zeros",
      "config": {}
    },
    "filters": 12,
```

```
      "kernel_constraint": null,
      "activation": "relu",
      "trainable": true,
      "kernel_size": [
        3
      ],
      "padding": "valid",
      "activity_regularizer": null,
      "kernel_initializer": {
        "class_name": "RandomNormal",
        "config": {
          "seed": null,
          "mean": 0.0,
          "stddev": 0.05
        }
      },
      "use_bias": true,
      "dilation_rate": [
        1
      ],
      "kernel_regularizer": null
    }
  },
  {
    "class_name": "MaxPooling1D",
    "config": {
      "pool_size": [
        2
      ],
      "data_format": "channels_last",
      "trainable": true,
      "padding": "valid",
      "name": "max_pooling1d_6",
      "strides": [
        2
      ]
    }
  },
  {
    "class_name": "Flatten",
    "config": {
      "trainable": true,
      "name": "flatten_2",
```

```
      "data_format": "channels_last"
    }
  },
  {
    "class_name": "Dense",
    "config": {
      "units": 20,
      "bias_regularizer": null,
      "kernel_regularizer": null,
      "name": "dense_6",
      "bias_constraint": null,
      "bias_initializer": {
        "class_name": "Zeros",
        "config": {}
      },
      "activation": "relu",
      "trainable": true,
      "activity_regularizer": null,
      "kernel_initializer": {
        "class_name": "RandomNormal",
        "config": {
          "seed": null,
          "mean": 0.0,
          "stddev": 0.05
        }
      },
      "use_bias": true,
      "kernel_constraint": null
    }
  },
  {
    "class_name": "Dense",
    "config": {
      "units": 6,
      "bias_regularizer": null,
      "kernel_regularizer": null,
      "name": "dense_7",
      "bias_constraint": null,
      "bias_initializer": {
        "class_name": "Zeros",
        "config": {}
      },
      "activation": "softmax",
```

```
            "trainable": true,
            "activity_regularizer": null,
            "kernel_initializer": {
              "class_name": "RandomNormal",
              "config": {
                "seed": null,
                "mean": 0.0,
                "stddev": 0.05
              }
            },
            "use_bias": true,
            "kernel_constraint": null
          }
        }
      ],
      "name": "sequential_3",
      "build_input_shape": [
        null,
        30,
        6
      ]
    },
    "keras_version": "2.2.4",
    "backend": "tensorflow"
}
```

ПРИЛОЖЕНИЕ И

```
import serial
import numpy as np
from sklearn.externals import joblib
import tensorflow as tf
from keras.models import model_from_json
from pynput.keyboard import Key, Controller

port = "COM9"
speed = 115200
ser = serial.Serial(port, speed, timeout=None)

keyboard = Controller()


gestures_to_keys = {
    "hand_left": Key.left,
    "hand_right": Key.right,
    "hand_up": Key.up,
    "hand_down": Key.down,
    "hit_forward": Key.esc,
    "hit_backward": Key.f5
}

jfile = open("model_conv.json", "r")
loaded_json = jfile.read()
jfile.close()
loaded_model = model_from_json(loaded_json)

loaded_model.load_weights("weights_conv.hdf5")
last_predict_date = 0
loaded_model.compile(loss='categorical_crossentropy',    optimizer='adam',
metrics=["accuracy"])

loaded_model._make_predict_function()
graph = tf.get_default_graph()

encoder = joblib.load("encoder.pkl")

vals = [[0 for j in range(6)] for i in range(30)]
```

```
        last_predictions = []
        threshes = [0.5, 0.5, 0.5, 0.95, 0.8, 0.65]
        last_predictions_size = 25  # 20
        last_cd_range = 14
        last_counter = None

        while True:
            line = ser.readline().decode().strip()
            values = line.split()

            if len(values) == 6:
                values = [float(i) for i in values]
                vals.append(values)

                last = np.array([vals[-30:]])
                with graph.as_default():
                    if last_counter is None:
                        current_prediction   =   loaded_model.predict(last)      #
вероятности классов
                        last_predictions += current_prediction.tolist()
                        last_predictions           =           last_predictions[-
last_predictions_size:]

                        last_probs = [0 for i in range(6)]
                        for j in range(6):
                            for i in range(len(last_predictions)):
                                last_probs[j]    +=    last_predictions[i][j]    /
last_predictions_size

                        # если трешхолд больше, чем вероятность, то жест не
предсказан
                        difs = [threshes[i] - last_probs[i] for i in range(6)]
                        min_dif = min(difs)
                        if min_dif <= 0:
                            current_gesture = difs.index(min_dif)
                            label                                                =
encoder.inverse_transform([current_gesture])[0]
                            print(label)
                            keyboard.press(gestures_to_keys[label])

                            last_counter = 0
                    elif last_counter is not None and last_counter < last_cd_range:
                        last_counter += 1
```

```
            last_predictions = []
        elif last_counter is not None and last_counter >= last_cd_range:
            last_counter = None
            last_predictions = []

    vals = vals[-30:]
```