Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«___»_____2024 г.

# БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Система выявления симптома замирания походки
у пациентов с болезнью Паркинсона

| | | | |
|---|---|---|---|
| Руководитель | _____ *подпись дата* | доцент, канд. техн. наук *должность, ученая степень* | Н.Ю. Сиротинина инициалы, фамилия |
| Выпускник | _____ *подпись дата* | | Е.А. Липатов инициалы, фамилия |
| Нормоконтролер | _____ *подпись* дата | доцент, канд. техн. наук *должность, ученая степень* | Н.Ю. Сиротинина инициалы, фамилия |

Красноярск 2024

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Болезнь Паркинсона – самое распространённое нейровегетативное расстройство. Это заболевание, распространённое у людей пожилого возраста, проявляющееся, прежде всего, в нарушении двигательных функций. Вероятность появления заболевания повышена у мужчин и людей, на которых воздействовали различные неблагоприятные факторы окружающей среды. Причины появления болезни Паркинсона не обнаружены. Заболевание при этом достаточно распространено [1]. Наибольшая вероятность появления заболевания у людей, чьи родственники уже подвержены этому заболеванию.

С каждым годом число людей в возрасте наиболее подверженном риску появления Болезни Паркинсона растёт. В 2015 году население старше 60 лет составляло чуть больше 912 миллионов. В 2021 году число людей старше 60 лет составляет около 1 083 211 000 человек. Это приблизительно, 14% населения всей Земли [2]. В 2020 году в мире насчитывалось от 7 до 10 миллионов людей с болезнью Паркинсона [3]. При этом эффективных методов лечения этого заболевания не известно. Используемая в настоящее время терапия не останавливают нейродегенерацию, она направлена на борьбу с симптомами и повышение качества жизни пациентов.

У заболевания есть множество симптомов. Один из самых пагубно влияющих на жизнь человека симптомов – это замирание походки. При проявлении этого симптома во время ходьбы человек замирает и не может передвинуть ногу вперёд, несмотря на намерение это сделать. В основном это явление длится несколько секунд; оно может сопровождаться потерей равновесия и травмами. Один из способов помочь человеку в борьбе с этим симптомом – подать в момент замирания некоторое воздействие, такое как звук или вибрация. Обнаружение симптома замирания походки в основном выполняется в виде тестов и опросников. Этим способом можно выявить факт наличия симптома, но нельзя на него воздействовать.

Для улучшения качества жизни пациента актуальной является задача выявления эпизода замирания походки непосредственно в момент возникновения. Это позволит врачам повысить эффективность диагностики и подбора лечения, а также позволит своевременно бороться с симптомом, подовая ритмичный сигнал во время эпизода замирания походки, улучшая походку.

**Целью работы** является разработка системы выявления симптома замирания походки у пациентов с болезнью Паркинсона.

**В ходе достижения поставленной цели должны быть решены следующие задачи:**

1. изучение предметной области, аналитический обзор аналогов;
2. выбор метода решения поставленной задачи;
3. проектирование системы, выбор аппаратного и инструментального ПО;
4. реализация и тестирование системы.

**В ходе достижения поставленной цели следующие получены следующие результаты:**

1. разработана и обучена модель нейронной сети для обнаружения замирания походки;
2. разработано мобильное приложение, выявляющее замирание походки и подающее стимулирующее воздействие.

Результаты представлены на конференции «Проспект Свободный», что подтверждается сертификатом (Приложение А), и планируется к опубликованию в сборнике материалов конференции.

# 1 Аналитический обзор подходов к распознаванию симптома замирания походки

В данной главе приведена краткая информация о проведённом анализе такой предметной области, как симптом заболевания Паркинсона – замирание походки. Приведёт обзор и анализ систем, выявляющих этот симптом. Сделан вывод по результатам этого анализа. Определенны наиболее подходящие алгоритмы для выявления Замирания походки, а также аппаратный вид системы.

## 1.1 Обзор предметной области

Болезнь Паркинсона встречается преимущественно у людей пожилого возраста. В промышленно развитых странах страдают от заболевания Паркинсона 3% людей старше 80 лет и 1% людей старше 60 лет. Наиболее важный фактор риска появления Болезни Паркинсона, это наличие близких родственников, у которых была замечена эта болезнь. Причины появления заболевания не выявлены, однако обнаружена зависимость от генетики. Так же вероятность появления заболевания повышена у мужчин и людей, на которых воздействовали негативные факторы, такие как воздействие пестицидов, употребление некачественной воды, злоупотребление алкоголем [1]. Заболевание сопровождается дегенерацией и гибелью нейронов, что приводит к нарушению двигательных функций [4].

Имеющиеся в настоящее время методы лечения являются симптоматическими и не останавливают нейродегенерацию.

Симптомами Паркинсона могут быть тремор, отсутствие реакции на стимулы, замедление активных и “содружественных” движений, невозможность произвольных движений, а также затруднение с удержанием равновесия. Среди прочих симптомов есть нарушение походки, то есть изменение характера ходьбы, нарушение речи, гипомимия, то есть снижение произвольных и

непроизвольных движений лицевой мускулатуры, изменение движения глаз, замирание походки [5].

Замирание походки (Frizzing of Gait, FOG) – симптом, при котором человек не может передвинуть ногу вперёд, несмотря на намерение сделать шаг. Это явление зачастую длится около нескольких секунд, однако это явление может продлиться куда дольше, доходя до того, что человеку требуется стимул, для возобновления ходьбы [6]. Этот симптом является одним из инвалидизирующих, то есть этот симптом крайне пагубно сказывается на жизни человека из-за частых падений. Методы, выявляющие замирание походки, основаны на тестах и опросниках [3].

В исследовании Evidence for Subtypes of Freezing of Gait in Parkinson's Disease [7], было предложено существование групп среди людей, у которых проявляется симптом замирания походки. Эти группы смогли разделить по типу действия, при котором возникает эпизод симптома. Удалось выделить группу людей, у которых симптом проявляется при моторном действии, таком как поворот. У людей в когнитивной группе может возникать замирание походки, к примеру, при выполнении нескольких дел одновременно. А у людей из лимбической группы симптом может возникнуть при тревоге.

При появлении явления запирания ходьбы у пациентов наблюдается вариабельность походки, значительное уменьшение длины шага и дрожание ног, а также изменение частоты шаговых движений. В то время как нормальная походка имеет основную частоту от 0,5 до 3 Гц, при возникновении замирания походки она снижается до 6 – 8 Гц [3], [8].

В последние годы выявление эпизодов замирания походки с помощью носимых датчиков становится всё более распространённым. В сочетании с методами машинного обучения такой подход позволяет куда быстрее и точнее определять эпизоды замирания походки. Благодаря этому возможно собирать данные для оптимальной терапии, а также позволяют следить за развитием симптома. Помимо этого, такая система способна улучшить ходьбу пациента подавая ритмичнее сигналы при возникновении эпизода замирания походки [3].

С развитием систем глубокого обучения они были внедрены в системы для распознавания человеческой деятельности. В области распознавания движений особенно хорошо себя проявило сочетание сверхточных нейронных сетей и сетей с долгой краткосрочной памятью [3].

Для сбора данных в различных исследованиях используются датчики различных типов. Чаще всего используется акселерометры, но применялись так же датчики электроэнцефалографии и плантарного давления. Часто в статьях, посвящённых обнаружению замирания походки, применятся несколько датчиков. Например акселерометры, гироскопы и магнитометры. Расположение устройства так же не одинаково. Преимущественно датчики располагают на различных частях ноги, но применялось и расположение в районе поясницы [9].

### 1.2 Требования к разрабатываемой системе

Проектируемая система должна соответствовать нескольким требованиям. Для удобства описания критериев разделим их на требования к аппаратной и к программной составляющей.

Аппаратная часть должна иметь в составе датчики для получения информации о движениях владельца системы. Таким датчиком может быть как минимум один датчик-акселерометр.

Для подачи стимула, купирующего эпизод замирания походки, она должна обладать устройством для подачи такого воздействия, например, динамиком или вибромотором.

Система должна быть удобной для повседневного использования, соответственно она должна быть компактной, лёгкой. Система должна быть достаточно дешёвой и распространённой.

Программная часть должна обрабатывать входящие данные с датчиков и определять, когда возникает эпизод замирания походки.

В случае возникновения эпизода замирания походки система должна инициировать подачу стимулирующего воздействия.

Помимо основной функции, система должна сохранять данные с датчиков и фиксировать в базе данных информацию об эпизоде замирания походки: момент возникновения и продолжительность.

Собранные данные могут использоваться для более точной настройки системы диагностики, возможного определения причины или предшествующих возникновению эпизода событий, а также в дальнейшем для попытки спрогнозировать эпизод и предотвратить его до проявления.

Система должна обеспечивать достаточно оперативную обработку получаемых данных для своевременной подачи стимулирующего сигнала, в случае возникновения эпизода замирания походки. Так как эпизод замирания походки, как правило, длится от 2 до 5 секунд, а в процессе ходьбы возможны естественные паузы, оптимальная задержка от начала эпизода замирания походки до генерации стимулирующего воздействия составляет 1-2 секунды.

Учитывая распространённость заболевания среди пожилых людей, программная часть должна обладать простым в использовании интерфейсом на русском языке.

### 1.3 Обзор существующих решений

Проведем обзор существующих решений, чтобы выявить перспективные методы определения эпизода замирания походки, возможные способы программной реализации и аппаратной составляющей системы, а также найти доступные наборы данных для обучающей выборки разрабатываемой системы.

В исследовании [3] 2020 года для тестирования использовались несколько наборов данных. В первом наборе данных использовался набор созданных вручную признаков. Второй набор данных состоял из мел-кепстральных коэффициентов (MFCCS), адаптированных к инерционным сигналам. Третий набор – спектральное представление данных на основе быстрого преобразования Фурье. Четвёртый набор основан на суммировании текущего спектрального окна и до трёх предыдущих окон анализа. Данные о движениях были собраны с

помощью носимого инерционного датчика IMU (Inertial Measurement Unit), расположенного на талии. В качестве данных на вход системы подаётся набор из трёх сигналов с акселерометра с частотой дискретизации 40Гц. Для оконной обработки сигнала брались окна длительностью между двумя и четырьмя секундами. Результаты работы показали, что наиболее эффективной оказалась рекуррентная нейронная сеть долгой краткосрочной памяти и носимый трехосный датчик ускорения датчик на поясе. В качестве данных для обучения лучше всего себя показали спектральные окна. Хотя данное решение и выполняет требования по работе системы, у него есть несколько минусов, связанных с датчиками системы. Для работы системы человек должен носить несколько датчиков на разных частях тела, и, соответственно, получить специально разработанные носимые устройства. Комплектующие системы иностранного производства, интерфейс программного обеспечения реализован на английском языке, что усложняет его использование.

В исследовании [9] так же рассматривается различные решения, и по результатам анализа это исследования можно увидеть, что среди множества прочих существующих решений, чувствительность (sensitivity) может равняться 0,63, специфичность (specificity) – 0,59, доля правильных ответов (accuracy) – 0,71, AUC – 0,76, F1-score – 0,78.

В исследовании [10] 2012 года, система состояла из смартфона и носимых акселерометров. Приложение состоит из двух компонент: онлайн и оффлайн. Классификатор обучатся в автономном режиме на заранее собранных и размеченных данных. В онлайн режиме происходит классификация окон. При возникновении замирания система подаёт ритмичные сигналы, для стимуляции возобновления походки. Для классификации окон проверялись следующие методы машинного обучения: случайные деревья (Random Tree, RT), случайные леса (Random Forest, RF), деревья решений и обрезанные деревья решений (C4.5), наивный байесовский классификатор (Naive Bayes, NB), Сети Байеса (BN), k-ближайших соседей (K-nearest neighbors, KNN), многослойный перцептрон (Multi-Layer Perceptron, MLP), AdaBoost и ансамблевый метаалгоритм

бэггинг (Bootstrap AGGregation, bagging) с обрезанными деревьями решений. Результаты анализа решений показали, что лучше всего с задачей справляются случайные деревья и AdaBoost с обрезанными деревьями решений. В дальнейшей работе авторы использовали случайные деревья из-за точности и простоты. Итоговая система смогла обнаружить события замирания походки со средней чувствительностью и специфичностью более 0,95 и средней задержкой обнаружения 0,34с. При обнаружении замирания походки система подаёт звуковой и вибрационный сигнал. Данная система наиболее близка к выполнению поставленных требований, однако в этом случае помимо телефона требуется носимый датчик. Так же приложение не поддерживает русский язык.

В исследовании [11] 2020 года система получает данные со смартфона, находящегося в сумке у пояса, и предоставляет информацию о наличие и продолжительности эпизодов FOG. В ходе работы были испытаны такие методы, как k-ближайших соседей, метод опорных векторов (Support Vector Machine, SVM), логистическая регрессия, кроме того, был рассмотрен подход с использованием искусственных нейронных сетей. В ходе тестов был как модель, была выбрана комбинация квадратичного SVM и k-ближайших соседей. Оценка разработанной системы получилась следующей: чувствительность (sensitivity) 0,95, специфичность 0,98, доля правильных положительных ответов (precision) 0,93 и доля правильных ответов (accuracy) 0,98. Минусом данной системы также является то, что она не поддерживает русского языка.

В исследовании [12] 2021 года, разработана трость, которая генерирует звуковые и тактильные сигналы, чтобы помочь пациентам преодолеть проблему возникновения FOG. На икрах закрепляется манжета для подачи стимулирующего сигнала во время эпизода FOG. Так же разработано мобильное приложение для отображения измерений пульса пациента, а также его точного местоположения в случае падения. Хотя это устройство и просто в использовании, его минусом является очевидная сложность аппаратного решения, ведь такое устройство как трость сложно производить массово, и, вероятно, придётся разрабатывать по частным заказам.

В исследовании [13] описываются наборы признаков для обнаружения замирания походки при использовании рекуррентных нейронных сетей. В этой работе утверждается, что признаки частотной области наиболее информативны. При этом доля правильных ответов составила до 0,93, специфичность до 0,9 и чувствительность до 0,81. В данном решении описан только алгоритм работы, а как таковое устройство не проектировалось.

В статье [14], 2019 года система использует три трехосных акселерометрических датчиков, надеваемых на спину, бедро и лодыжку. Система пытается предсказать возникновение FOG. Система классифицирует три класса событий: pre-FoG, no-FoG и FoG. При обнаружении эпизода замирания походки система генерирует ритмичные сигналы, используя носимое устройство. Данная система, как и несколько предыдущих, сложна из-за нескольких носимых датчиков и труднодоступна обычному пользователю.

Результаты проведенного сравнения аналогов представлены в таблице 1.

Таблица 1 – Сравнение подходов к выявлению симптома замирания походки

| **Решение** | **Luis S. 2020** | **Sinziana M. 2012** | **Luigi B. 2020** | **Joelle H. 2021** | **Florenc D. 2019** |
|---|---|---|---|---|---|
| Датчики | Датчик-акселерометр | Три датчика-акселерометра | Датчик-акселерометр смартфона | Датчик акселерометр, датчик пульса | Три датчика-акселерометра |
| Стимул | - | Звук, вибрация | - | Звук, вибрация, свет | Вибрация |
| Эргономичность | Трёхосевой датчик-акселерометр на поясе | Смартфон и носимые датчики | Смартфон в поясной сумке | Устройство располагается в трости | Смартфон и датчики на спине, бедре и лодыжке |
| Доступность | Интерфейс на английском языке | Интерфейс на английском языке | Интерфейс на английском языке | Интерфейс на английском языке | Интерфейс на английском языке |
| Запись данных | + | + | + | + | + |
| Доля правильных ответов | Нет данных | Нет данных | 0,983 | Нет данных | Нет данных |
| Специфичность | Нет данных | 0,95 | 0,988 | Нет данных | Нет данных |
| Чувствительность | Нет данных | 0,95 | 0,954 | Нет данных | Нет данных |
| Быстродействие | Нет данных | 0.34 сек. | 1-2 сек. | Нет данных | 2-4 сек. |
| Метод выявления | LSTM | AdaBoost | SVM и KNN | Нет данных | Нет данных |

Сравнение подходов к выявлению симптома замирания походки показало, что все решения используют для получения информации о передвижениях датчик-акселерометр. Среди стимулирующих воздействий решения отдают предпочтения звуковым и вибрационным воздействиям. Так же можно отметить, что решения применяли методы выявления замирания походки, основанные на различных принципах. Так же можно оценить различные метрики у рассмотренных решений.

### 1.4 Выводы по 1 главе

При анализе существующих решений мы выяснили, что ни одна из систем не удовлетворяет полностью требованиям задания. Все описанные решения имеют различные минусы. Самый основной из них – это сложность в получении такого решения рядовому пациенту, в виду того, что решения, как правило, выполнены в виде специально разработанных носимых устройств и реализованы на импортном оборудовании. Поэтому принято решение разрабатывать собственную систему.

По результатам обзора определено, что для определения эпизодов замирания походки в настоящее время используют алгоритмы глубокого и машинного обучения. Наилучшие результаты получены при использовании рекуррентных нейронных сетей, это определило выбор их в качестве базы для реализации разрабатываемой системы.

Наиболее привлекательным для реализации аппаратной части системы из рассмотренных в обзоре вариантов представляется использование в качестве носимого устройства смартфона. Смартфоны в настоящее время достаточно доступны и распространены, пожилые люди в основном владеют базовыми навыками их использования. Смартфоны, как правило, имеют встроенный акселерометр, который может использоваться в качестве датчика. В рассмотренных работах подтверждается возможность получения требуемых данных со смартфона, носимого в поясной сумке. Смартфон также имеет

вибросигнал и звуковой сигнал, что позволяет подавать стимулирующие воздействия для снятия симптома замирания походки.

Обзор показал, что большинство решений помимо основных функций ведут запись информации об эпизодах замирания походки, которые используются в дальнейшем для оценки состояния пациента и корректировки курса лечения. Данная функция представляется полезной и будет реализовываться в разрабатываемой системе.

## 2 Проектирование системы

Для проектирования и описания общей структуры системы воспользуемся шаблоном проектирования C4. Для описания пользовательского интерфейса используем диаграммы пригодности прецедентов. Для описания классификатора используем диаграмму последовательностей. Так же опишем выбранную модель нейронной сети и выберем инструменты разработки.

### 2.1 Общая структура системы

В качестве аппаратной части системы принято решение использовать смартфон и его датчики. Смартфоны на данный момент получили достаточное распространение, чтобы считать это устройство легкодоступным, а устройство их операционной системы в большинстве своём достаточно интуитивно понятно для наших целей. Что касается способа получения информации о движениях пользователя – практически все смартфоны снабжены необходимым нам датчиком-акселерометром, а сам смартфон достаточно удобен для повседневного ношения.

Программная часть системы реализуется как мобильное приложение.

В качестве целевой операционной системы выбрана ОС Android.

Сервисная часть приложения основная. Она работает в фоновом режиме постоянно и получает данные с датчика-акселерометра смартфона. Полученные данные она обрабатывает в модели. От неё же сервис получает информацию о том, происходит ли эпизод замирания походки. В случае если он происходит, приложение подаст настроенный в клиентском приложении стимул. Так же система запишет данные с датчика акселерометра, полученные до появления эпизода, а также время появления эпизода и его длительность.

Классификатор, выявляющий замирание походки, реализует нейросеть, которая классифицирует полученные данные и выдаёт информацию о том, происходит ли в данный момент эпизод замирания походки. По результатам

анализа аналогичных решений мы выяснили, что в задачах обнаружения замирания походки лучше всего показала себя рекуррентная нейронная сеть, так что в серверной части приложения используем её. Она будет получать данные с датчика-акселерометра клиентского приложения, и отправлять ему результат с информацией, происходит ли в данный момент времени эпизод замирания походки.

Диаграмма «Контекст» используется, для демонстрации того, как проектируемая система взаимодействует с окружением. Данная диаграмма представлена на рисунке 2.1.


Рисунок 2.1 – Диаграмма «Контекст»

Диаграмма «Контейнер» демонстрирует верхнеуровневую архитектуру приложения. Данная диаграмма представлена на рисунке 2.2.

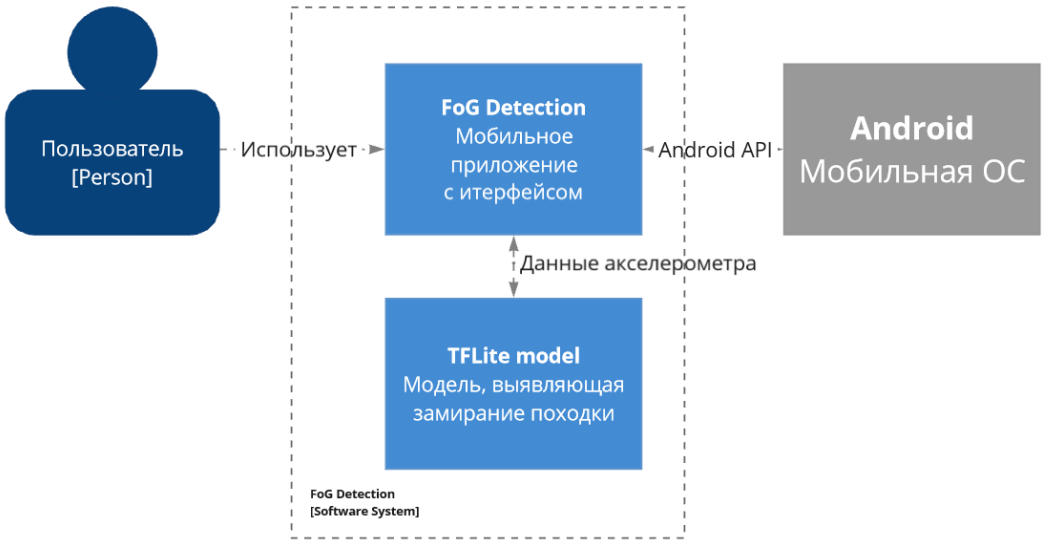


Рисунок 2.2 – Диаграмма «Контейнер»

Диаграмма «Компонент» — это детальное представление архитектуры внутри одного контейнера. Так как устройство конвертированной модели сокрыто в реализации библиотеки, нас интересует только один контейнер. Его диаграмма представлена на рисунке 2.3.

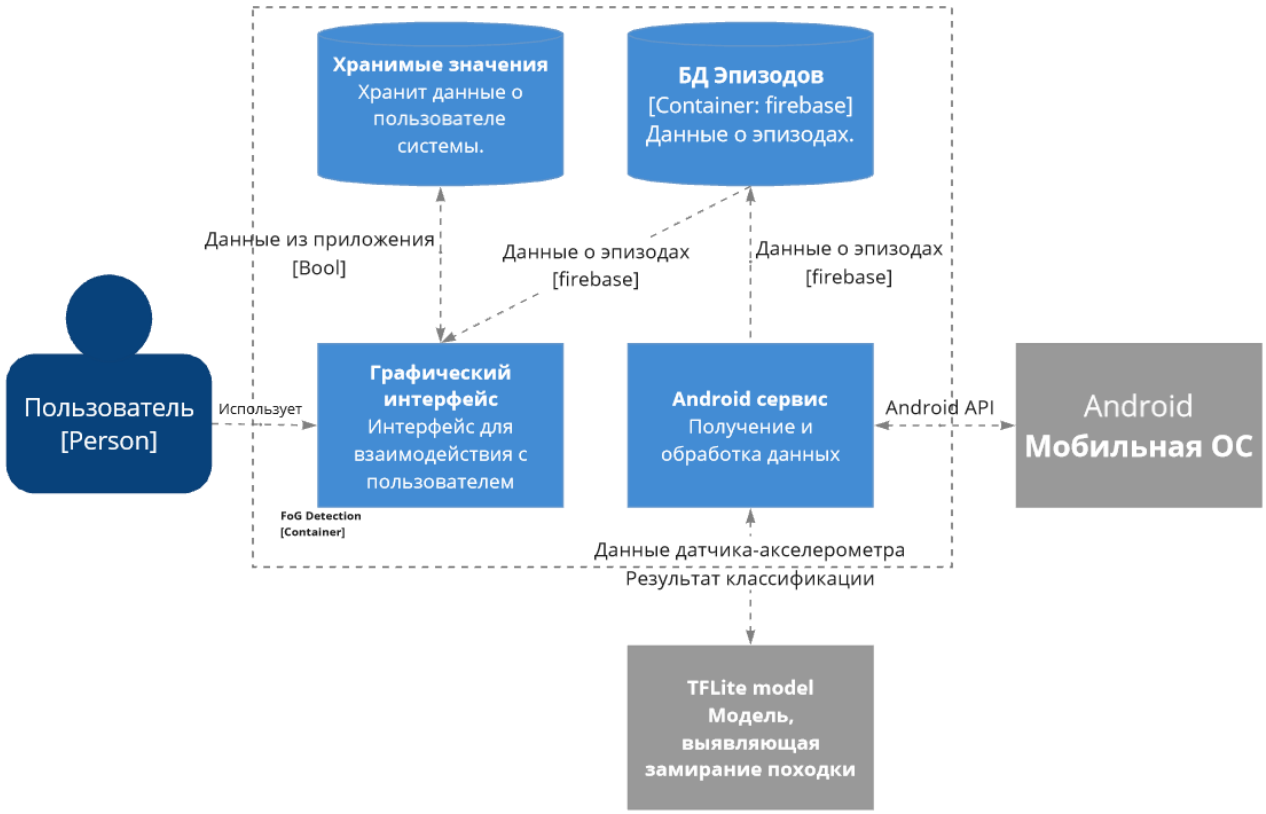


Рисунок 2.3 – Диаграмма «Компонент» для мобильного приложения

Приложение получает данные с датчика-акселерометра, анализирует их и при необходимости создаёт стимулирующее воздействие, обращаясь к соответствующим аппаратным устройства телефона, используя Android API.

Сервис получает данные с датчика-акселерометра и отправляет их серверному приложению пакетами длинной в 4 секунды. Серверное приложение отправляет на сервис сигнал о том, что в пришедший участок времени происходит эпизод замирания походки. Если пришёл сигнал, то сервис запускает стимулирующее воздействие. Сервис заносит в базу данных информацию, о каждом участке времени длинной в 4 секунды. Приложение обращается к базе данных, для отображения результатов в графическом интерфейсе.

## 2.2 Диаграммы пригодности

Для представления архитектуры пользовательского интерфейса программного обеспечения, разработаем диаграммы пригодности для каждого прецедента. В этих диаграммах выделим несколько видов сущностей: объекты, соответствующие главным элементам интерфейса, объекты-сущности, задающие базу дынных или файлы, и процессы, необходимые для обозначения функций или классов.

### 2.2.1 Прецедент «Настройка программы»

Пользователь может настроить приложение, выбрав язык отображаемого текста и тип стимулирующего воздействия, которое будет подавать система, в случае обнаружения эпизода замирания походки. Для этого он на главном экране переходит в окно «Настройки», где он настраивает вышеуказанные параметры. После чего он может вернуться на главный экран.

Диаграмма пригодности прецедента изображена на рисунке 2.4.


Рисунок 2.4 – Диаграмма пригодности прецедента настройка программы

**Название прецедента:** Настройка программы.

**Цель сценария:** получить необходимые для работы программы параметры.

**Предусловия:** открыто «Домашнее окно».

**Основной сценарий:**

1. пользователь переходит в окно настроек;
2. пользователь выбирает настройки;
3. данные сохраняются в БД;
4. пользователь возвращается в «Домашнее окно».

**Постусловия:** параметры настроек сохраняются.

### 2.2.2 Прецедент «Просмотр базы данных»

Пользователь на главном экране переходит в окно «Записи». Их этого окна он может вернуться, нажав кнопку «Назад». Диаграмма пригодности прецедента изображена на рисунке 2.5.


Рисунок 2.5 – Диаграмма пригодности прецедента «Просмотр базы данных»

**Название прецедента:** Просмотр базы данных.

**Цель сценария:** просмотреть записи о замираниях походки.

**Предусловия:** открыто «Домашнее окно».

**Основной сценарий:**

1. пользователь переходит в окно просмотра эпизодов;

2. пользователь возвращается в «Домашнее окно».

### 2.2.3 Прецедент «Включение распознавания»

Для того, чтобы программа начала функционировать, пользователь должен включить её на главном экране.

Диаграмма пригодности прецедента изображена на рисунке 2.6.

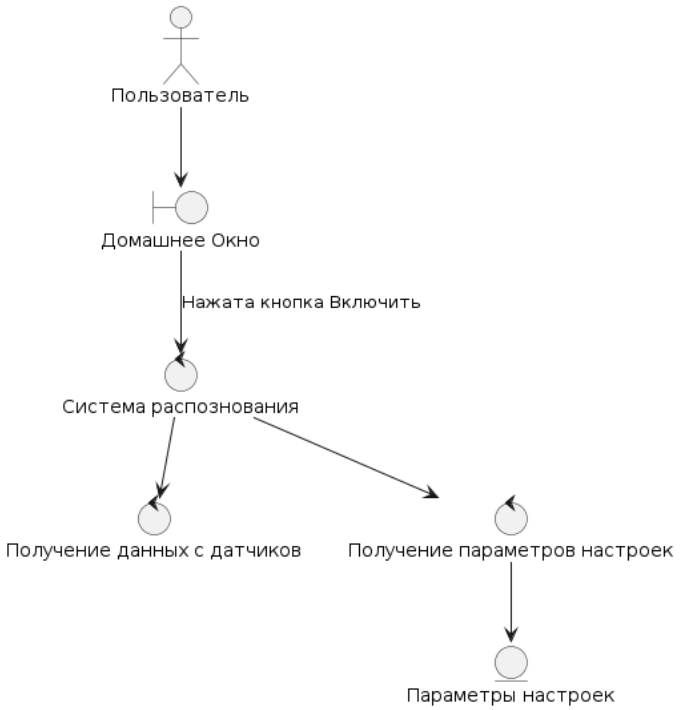


Рисунок 2.6 – Диаграмма пригодности прецедента «Включение распознавания»

**Название прецедента:** Включение распознавания.

**Цель сценария:** запустить распознавание эпизодов замирания походки.

**Предусловия:** открыто «Домашнее окно».

**Основной сценарий:**

1. пользователь нажимает кнопку «Включить»;

2. запускается фоновый сервис;

3. фоновый сервис получает данные о пользователе из базы данных пользователя;

4. фоновый сервис начинает получать данные с датчика-акселерометра устройства.

**Постусловия:** запускается фоновый сервис.

### 2.3 Диаграмма последовательностей

Что бы лучше понимать принцип взаимодействия модулей программы, разработаем диаграмму последовательностей, для сценария, когда программа включена, и обнаруживает эпизод замирания походки. Диаграмма изображена на рисунке 2.7.

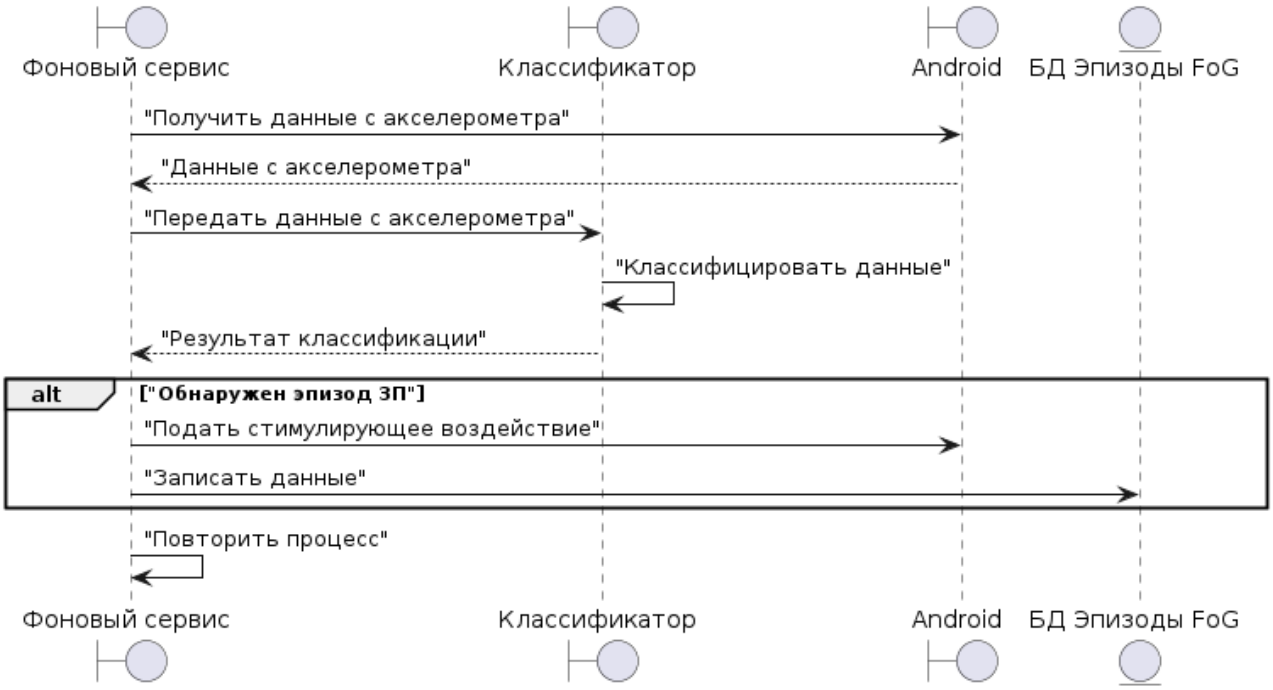


Рисунок 2.7 – Диаграмма последовательностей

### 2.4 Хранение данных

Для хранения данных об эпизодах замирания походки нам потребуется простейшая база данных. Реализуем её с помощью базы данных firebase.

Наша база данных тривиальна и представляет собой одну таблицу с полями даты, времени и данных датчика-акселерометра. Эта база данных хранит записи о времени и дате возникшего эпизода замирания походки, а также данные с датчика акселерометра за промежуток, в который возник эпизод.

Данные, вводимые пользователем в программе храниться, с использованием встроенных в android studio методов хранения простых полей.

### 2.5 Модель нейронной сети

Так как в результате обзора было определено, что наибольшей эффективностью в выявлении замирания походки обладают рекуррентные нейронные сети, остановимся на их модели подробнее.

Обычные нейронные сети не запоминают контекст работы. Каждый набор входных сигналов обрабатывается независимо от остальных. Это влечёт за собой проблемы в работе данных сетей при использовании их в областях, где такой контекст необходим, в частности, при работе с временными рядами. Рекуррентные нейронные сети призваны решить эту проблему.

Рекуррентные нейронные сети содержат в себе обратные связи. Пример нейрона классической рекуррентной нейронной сети изображён на рисунке 2.8.

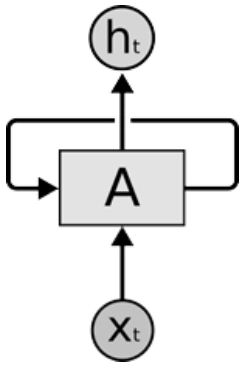


Рисунок 2.8 – Пример нейрона рекуррентной нейронной сети [15]

Наличие связи, направленной от нейрона к самому себе, описывает, что выход нейрона передаётся между шагами сети. То есть на новом этапе работы сохраняет предыдущий результат.

Задача обнаружения замирания походки представляет собой задачу анализа временного ряда. Распознающая модель получает на вход данные с акселерометра, и для понимания, происходит ли в данный момент времени за-

мирание походки, необходимо анализировать некий контекст предыдущих показаний. Поэтому для таких задач обычно выбираются рекуррентные нейронные сети. Несмотря на наличие своеобразной «памяти», у простейшей рекуррентной сети она слишком короткая для некоторых задач; кроме того, обучение простейших рекуррентных нейронных сетей затруднено из-за проблемы затухания/взрыва градиента. Для решения этой проблемы существуют нейронные сети долгой краткосрочной памяти (Long-Short Term Memory, LSTM). Модель нейрона такой сети представлена на рисунке 2.9.

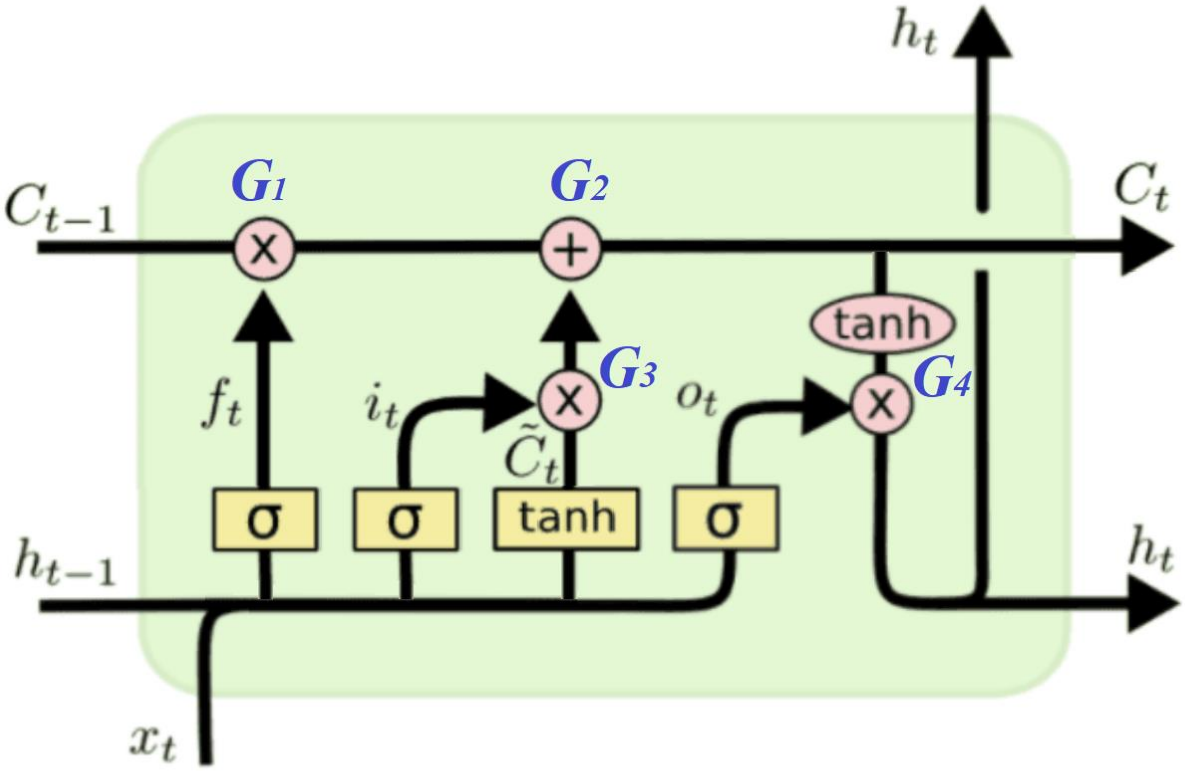


Рисунок 2.9 – Модель нейрона сети LSTM [15]

В данной модели нейрон представляет собой достаточно сложную конструкцию и содержит не один, а несколько слоёв. Разберём работу такой системы подробнее.

На рисунке 2.9 мы видим несколько обозначений. Стрелки обозначают передачу вектора сигналов. Кругами обозначены поточные операции – сложе-

ние и умножение векторов. Прямоугольники – внутренние слои нейрона LSTM. Надписи в прямоугольнике означают: σ – что слой реализует функцию сигмоиды; tanh – что слой реализует функцию гиперболического тангенса. В местах, где одна стрелка разделяется на несколько, происходит копирование данных. Там, где стрелки объединяются, происходит объединение данных.

Работа сети заключается в следующем. По верхней лини передаётся состояние ячейки. На этой линии находится два круга. Эти круги называются фильтрами.

Перемножающий фильтр позволяет удалить из памяти некоторые элементы. Как видно по схеме, на вход операции перемножения подаётся выход сигмоидального слоя. Он реализует функцию сигмоиды, которая позволяет выделить элементы, которые можно удалить из памяти. Практически сигмоидальный слой возвращает вектор значений от 0, до 1. Перемножение его с вектором состояния и позволяет удалять элементы. Сам сигмоидальный слой называют «Слоем фильтра забываний». Он получает на вход выход предыдущего входа нейрона, объединенный с действующим входным воздействием. Выход этого слоя следует интерполировать так, что 0, это полностью забытые данные, а 1 – это сохранённые данные.

Далее отрабатывает складывающий фильтр. На его вход подаётся результат работы конструкции, работу которой мы разберём позже. Эта конструкция, называемая «слой входного фильтра», позволяет определить, какие новые данные мы ходим «запомнить». Решение этой задачи реализуется несколькими слоями. Сигмоидальный слой, получающий такие же входные воздействия, определяет значения, которые необходимо обновить. Затем tanh-слой, реализующий гиперболическую функцию, строит вектор новых значений, которые должны быть добавлены в состояния. Перемножение выхода этих 2х слоёв и добавляется фильтром к вектору состояний.

Так же в системе есть ещё один фильтр. Этот фильтр и определяет, какую информацию мы получим на выходе. Тут так же сперва применяется сигмоидальный слой. Его устройство остаётся таким же. Он определяет какую

информацию мы будем выводить. Затем значения состояния проходят, через than-слой, формируя значения в диапазоне ] -1, 1[. Они перемножаются с выходом сигмоидального слоя, формируя выходные значения.

Обучение рекуррентных нейронных сетей осуществляется известным методом градиентного спуска с вычислением градиента методом обратного распространения ошибки. При этом цикличность процесса во времени раскладывается в многослойную нейронную сеть, в которой каждый скрытый слой соответствует временному шагу работы сети. При этом необходимо учитывать, что все «слои» этой нейронной сети используют одну и ту же матрицу весовых коэффициентов. Поэтому для корректировки весов берется сумма градиентов по всем тактам функционирования («слоям» нейросети), и поправку к весовым коэффициентам считаем один раз для суммарного по всем временным тактам градиента.

### 2.6 Выбор инструментов

Для реализации приложения для операционной системы Android естественным выбором является среда Android Studio, имеющая необходимые библиотеки для реализации необходимого функционала, такого как обращения к аппаратным ресурсам мобильного устройства, для получения данных с датчика-акселерометра, и подачи стимулирующего воздействия звуковым и вибрационным сигналами, а также для отображения требуемых графических окон.

Для программирования мобильного приложения выбран язык Kotlin, из-за удобства написания на нём простых функций мобильного приложения.

Для написания модуля обнаружения симптома используем язык программирования Python, ввиду наличия большого числа удобных библиотек для реализации необходимого нам функционала.

Для написания нейросети выбрана библиотека TensorFlow, предоставляющая готовые шаблоны слоёв, в том числе для описания LSTM сетей.

Так же в работе используется библиотеки numpy для упрощения работы с математическими формулами и pandas для обработки данных датасета.

## 2.7 Выводы по главе 2

Для проектирования и описания общей структуры системы выбран шаблон проектирования C4.

В соответствии с выбранным шаблоном проектирования выполнено проектирование разрабатываемого приложения, а именно: построены диаграммы, описывающие разрабатываемую систему, построены диаграммы пригодности для прецедентов взаимодействия пользователя с приложением, построена диаграмма последовательностей, для описания работы фонового сервиса, не взаимодействующего с пользователем.

На основании обзора методов решения задачи обнаружения симптома замирания походки выбрана для реализации модель нейронной сети LSTM, изучены особенности ее функционирования и обучения.

Результаты, полученные на данном этапе, позволяют перейти к разработке приложения.

# 3 Реализация и тестирование системы

В данной главе описан процесс разработки системы. Так как работа с моделью нейронной сети и работа с мобильным приложением являются различными задачами, разобьём этапы разработки на 2 части: разработка модели нейронной сети, и разработка мобильного приложения.

## 3.1 Подготовка обучающей выборки

Для обучения нейронной сети нам потребуются классифицированные данные о возникновении эпизодов замирания походки, полученные с датчика-акселерометра или данные, которые можно к ним привести.

Обучающая выборка формируется на основе датасета, размещенного на сайте Kaggle [16] в 2023 году. Данные для датасета собирали три исследовательские группы: Центр изучения движения, познания и подвижности Института неврологии Тель-Авивского медицинского центра Сураски (инициатор исследования), Исследовательская группа нейрореабилитации при Католическом университете Левена в Бельгии и Центр трансляционных исследований подвижности и падений при Институте старения Хинды и Артура Маркуса, связанный с Гарвардской медицинской школой в Бостоне. Датасет содержит данные с датчика-акселерометра расположенного в области пояса у людей с симптомом замирания походки. Данные представлены в формате csv.

Датасет включает в себя несколько таблиц:

- tDCS FOG – данные, собранные в лаборатории, когда испытуемые выполняли протокол, провоцирующий эпизод замирания походки;

- DeFOG - данные, собранные дома, когда испытуемые выполняли протокол, провоцирующий эпизод замирания походки;

- Daily Living – набор данных, состоящий из одной недели непрерывных круглосуточных записей от шестидесяти пяти испытуемых. У сорока пяти испытуемых наблюдаются симптомы замирания походки, и у них также есть

серии в наборе данных DeFOG, в то время как у других двадцати испытуемых симптомы замирания походки не проявляются, и у них нет серий в других источниках данных.

Таблицы разбиты на несколько частей.

**train/** – папка, содержащая серии данных из обучающего набора в трех вложенных папках: tdcsfog/, defog/ и notype/. Серии в папке notype взяты из набора данных defog, но не имеют аннотаций событийного типа. Поля, присутствующие в этих сериях, зависят от папки.

**test/** – папка, содержащая серии тестовых данных, в которых предоставляются только поля Time, AccV, AccML и AccAP.

**unlabeled/** – папка, содержащая неаннотированные серии данных из ежедневного набора данных, по одной серии на испытуемого. Сорок пять испытуемых также имеют серии в наборе данных defog, некоторые из них в тренировочном и некоторые в тестовом наборе. Данные акселерометра имеют единицы измерения g.

**tdcsfog_metadata.csv** – таблица, идентифицирующая каждую серию в наборе данных tdcsfog по уникальному условию субъекта, посещения, теста, лекарства.

**defog_metadata.csv** – таблица, идентифицирующая каждую серию в наборе данных defog по уникальному условию Subject, Visit, Medication.

**daily_metadata.csv** – таблица, содержащая серии в ежедневном наборе данных, которые идентифицируется по идентификатору субъекта. Этот файл также содержит время начала записи.

**subjects.csv** – таблица метаданных для каждого субъекта исследования, включая его возраст и пол, а также: посещение, для таблиц daily и defog, число лет с момента диагностирования Заболевания Паркинсона, Оценка по Унифицированной шкале оценки болезни Паркинсона во время приема/отмены препарата, оценка по опроснику Замирания Походки.

**events.csv** – таблица метаданных для каждого события Замирания Походки во всех сериях данных. Время события совпадает с метками в сериях данных.

**tasks.csv** – таблица метаданных задач для серий в наборе данных defog.

Данные, необходимые для обучения нейронной сети, находятся в таблицах tDCS FOG и DeFOG. Остальные данные могут быть полезны в дальнейшем для улучшения качества работы системы и дообучения нейронной сети.

Таблицы содержат следующие поля:

Time – поле временных меток по 1 за миллисекунду записи;

AccAP, AccML, AccV – поля, содержащие данные с датчика акселерометра по осям x, y и z соответственно, единицы измерения – метр на секунду в квадрате.

Отметим, что поле AccV, инвертировано, то есть его значения находятся в пределах от 9,81 до 0.

StartHesitation, Turn, Walking содержат данные о возникновении эпизода замирания походки одного из трех типов; они представлены в бинарном формате. Фрагмент данных из таблиц представлен в таблице 3.1.

Таблица 3.1 – Фрагмент таблицы DeFOG

| Time | AccV | AccML | AccAP | StartHesitation | Turn | Walking |
|---|---|---|---|---|---|---|
| 0 | -1,0232983 | 0,0392175 | 0,0562038 | 0 | 0 | 0 |
| 1 | -1,0235866 | 0,0401140 | 0,0593160 | 0 | 0 | 0 |
| 2 | -1,0262039 | 0,0409568 | 0,0605730 | 0 | 0 | 0 |

Для обучения модели нам необходимо загрузить данные их обеих таблиц. Для этого напишем код на языке python, используя библиотеку pandas.

Так как для каждого пациента, существует своя таблица, нам необходимо при загрузке добавлять к таблице столбик id, по имени таблицы и объединить данные из разных таблиц в одну. Для этого мы загружаем каждую таблицу

каждого пациента, загружаю вышеупомянутые поля, и объедением их в две таблицы – tdcsfog и defog соответственно.

Далее нам необходимо объединить полученные таблицы. А также обработать данные в итоговой таблице.

В исходной таблице содержатся данные об эпизодах замирания походки трех типов: Turn – при повороте, Walking – во время ходьбы, и StartHesitation – в начале движения. В нашей работе рассматривается задача идентификации эпизода замирания походки независимо от типа, поэтому эти три столбца объединяем в новый столбец Event. Данные в нём представлены в бинарном формате – 1, если хотя бы один из столбцов Turn, Walking, и StartHesitation, был равен 1, и 0, в противном случае.

Так как нам известно, что данные в таблице представлены в формате данных с датчика-акселерометра, то мы удаляем все строки, значения Acc в которых превышает ускорение свободного падения, поскольку это свидетельствует об ошибке в данных.

Так же в изначальном датасете данные с датчика акселерометра собраны с частотой в 100 Гц. Учитывая среднюю продолжительность эпизода, такая частота представляется избыточной. Для уменьшения размеров датасета отберём каждую 10 строчку исходной таблицы, таким образом уменьшив частоту до 10 Гц.

Для дальнейшей обработки выделяем список всех значимых для нас величин, то есть столбцов с информацией необходимой для обучения.

Код вышеупомянутой обработки таблицы приведён в приложении Б.

Так как данные для обучения являются данными с акселерометра, то их величины относительно уравновешены.  Для ускорения дальнейшей передачи данных в модель, нормализацию данных проводить не будем.

Далее, обучая модель, мы будем подавать в нее данные, группируя их по столбцу id, так что нам потребуется функция, которая будет разделять все данные в одной группе id, на данные train, validation, test. В соотношении 70%/20%/10%.

Для обучения модели исходные данные необходимо подавать в форме скользящих окон. Для этого создан класс, выполняющий эту задачу. Класс генератора окон получает на вход таблицы тренировочных, тестовых, и валидационных данных, а также целевое поле.

В конструкторе данного класса создаются параметры необходимые для разделения данных таблиц.

Класс реализует методы make_dataset и split_window.

Метод make_dataset формирует данные выборки в tf.data.Dataset. Для выделения окна данных он использует ещё один метод split_window.

Метод split_window выполняет выделение окон данных. Он формирует inputs и lables из данных окна, которые используются в make_dataset для преобразования временного ряда DataFrame в обучающую выборку в формате tf.data.Dataset, состоящую из пар (input_window, label_window).

Код конструктора этого класса и методов make_dataset и split_window представлены в приложении В.

### 3.2 Выбор функции потерь и целевой метрики

Для обучения модели нам необходимо выбрать функцию потерь и целевую метрику.

Поскольку задача идентификации эпизода замирания походки может быть сформулирована как задача бинарной классификации, в качестве функции потерь должна использоваться двоичная перекрестная энтропия (binary crossentropy).

Для оценки качества работы обученной модели необходимо выбрать целевую метрику качества.

В обучающей выборке классы представлены неоднородно, эпизоды возникновения замирания походки составляют незначительную часть от общего числа замеров, поэтому метрика «доля правильных ответов» (accuracy) не может быть использована.

Для оценки качества работы алгоритма на каждом из классов по отдельности используются метрики «точность» (precision) и «полнота» (recall).

Precision можно интерпретировать как долю правильно диагностированных среди всех диагностированных эпизодов замирания походки.

Recall показывает, какую долю эпизодов замирания походки из всех таких эпизодов, входящих в тестовую выборку, диагностировала модель.

В отличие от метрики accuracy, precision и recall не зависят от соотношения классов и потому применимы в условиях несбалансированных выборок. Перед нами стоит задача найти оптимальный баланс между этими двумя метриками, т.к. они, как правило, антагонисты: при улучшении одного показателя ухудшается другой. Поэтому среди метрик качества для данной задачи больше всего подходит F1-метрика, так как нам необходимо, чтобы модель, как можно точнее диагностировала возникновение симптома замирания походки, но при этом по возможности снизить частоту ложных срабатываний. Метрика F1 определяется по формуле:

$$F1 = 2 * \frac{precision * recall}{precision + recall}. \quad (1)$$

Метрика F1 достигает максимума (значения 1) при полноте и точности, равных единице, и близка к нулю, если один из аргументов близок к нулю.

Так как встроенной реализации F1-метрики в библиотеки tensorflow нет, для ее вычисления создана специальная функция. Код этой функции приведен в приложении Г.

### 3.3 Разработка модели нейронной сети

В качестве модели для распознавания выбрана нейронная сеть LSTM. В библиотеке tensorflow есть встроенная реализация LSTM слоёв. При разработке модели были рассмотрены существующие решения среди моделей выявления

симптома замирания походки, представленные на сайте Kaggle в соответствующей задаче [16]. Пример сети, найденной среди решений, представлен на рисунке 3.1.

```
model = Sequential(name='Prediction_Gait')
model.add(LSTM(80,input_shape=(lookback,len(all_features),),return_sequences=True))
model.add(LSTM(128,activation='relu'))

model.add(Dense(80,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))

model.add(Dense(10,activation='sigmoid'))

model.add(Dense(3,activation='softmax'))
```

Рисунок 3.1 – LSTM модель, представленная на ресурсе Kaggle [16]

Как мы видим, модель представляет собой 2 слоя LSTM и несколько полносвязных слоёв. Графическое изображение сети представлено на рисунке 3.2.

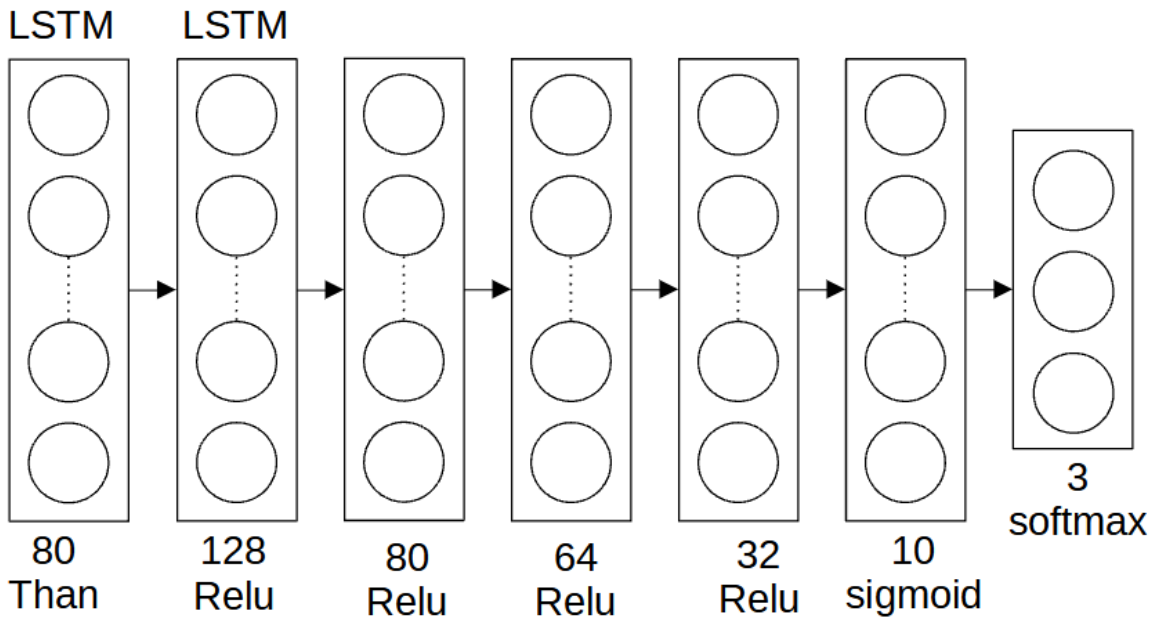


Рисунок 3.2 – Модель сети с Kaggle

Данная модель сети выполняла задачу класификации по 3м классам, так что для выполнения нашей задачи данную модель пришлось несколько изме-

нить. Так выходной слой сети имеет активацию softmax, что характерно для классификации по нескольким признакам. Наша классификация сведена к бинарной, так что функцию активации последнего слоя была заменена на sigmoid, более характерную для бинарной классификации. Так же изменена функция активации слоя LSTM с relu, на обыкновенную для такого слоя than. В нашей сети отличаются входные размерности, их изменим на соответствующие.

При попытке обучить такую модель, было обнаружено, что она избыточна, из-за чего быстро происходило переобучение и показатели метрики качества работы сети сильно снижалась. При попытке попытке избавться от переобучения добавлением слоёв dropout, было обнаружено, что такой модели требуется параметр dropout равный 0,9, для достижения метрики precision 0,4-0,5. После этого было решено модель упростить.

После перебора различных конфигураций сети, была составлена сеть с одним слоем LSTM, и одним полносвязным выходным слоем, а также слоем Dropout с параметром 0,5.

Код описания итоговой модели представлен на рисунке 3.3.

```
1 lstm_model = tf.keras.Sequential([
2                             tf.keras.layers.LSTM(80, activation='tanh', in-
  put_shape=input_shape),
3                             tf.keras.layers.Dropout(0.5),
4                             tf.keras.layers.Dense(1, activation='sigmoid')
5                             ])
```

Рисунок 3.3 – LSTM модель

Графическое представление спроектированной модели сети, составленное с помощью сайта netron [17], представлено на рисунке 3.4.

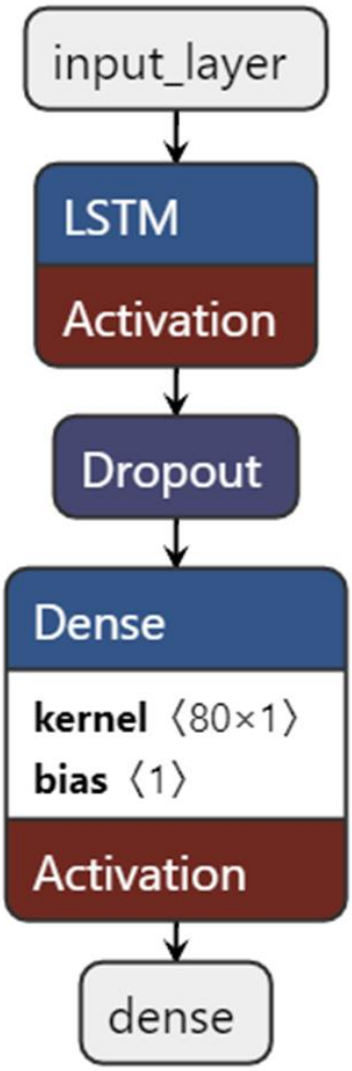


Рисунок 3.4 – Архитектура сети

### 3.4 Обучение модели

При разработке системы были рассмотрены различные варианты обучения модели.

В первом варианте данные не делились на пациентов, и передавались нейросети полной таблицей. Этот подход не дал подходящих результатов.

Было принято решение обучать модель на различных пациентах отдельно. Этот подход показал себя лучше.

Изначально данные одного пациента передавались модели так же целиком.

Альтернативой этому подходу стала передача данных модели окнами. Этот подход показал себя лучше всего, поэтому итоговая модель обучалась с использованием именно этого метода.

Окончательный вариант обучения системы состоит из следующих этапов.

1. Загружается исходный датасет (метод описан в параграфе 3.1).

2. Создается и компилируется модель нейронной сети (описана в параграфе 3.2). В качестве функции потерь указывается двоичная перекрестная энтропия (binary crossentropy), в качестве целевых метрик – F1-score, accuracy и precision; в качестве оптимизатора градиентного спуска выбираем стандартный оптимизатор Adam.

3. Выделяем из датасета фрагмент данных, относящий к одному пациенту.

4. Отделяем тестовый набор данных.

5. Формируем обучающую и валидационную подвыборки.

6. На основе обучающей выборки создаём окно данных (описано в пункте 3).

7. Выполняем цикл (шаг) обучения.

8. Если эпоха обучения не завершена, переходим к пункту 6; иначе оцениваем качество работы модели.

9. Если качество работы модели удовлетворительно или исчерпано максимальное число эпох обучения, обучение завершается. Если нет, переходим к пункту 5.

10. Выполняется итоговое оценивание качества работы модели на тестовом наборе данных.

Динамика изменения метрик на одном пациенте при обучении представлена на рисунке 3.5.

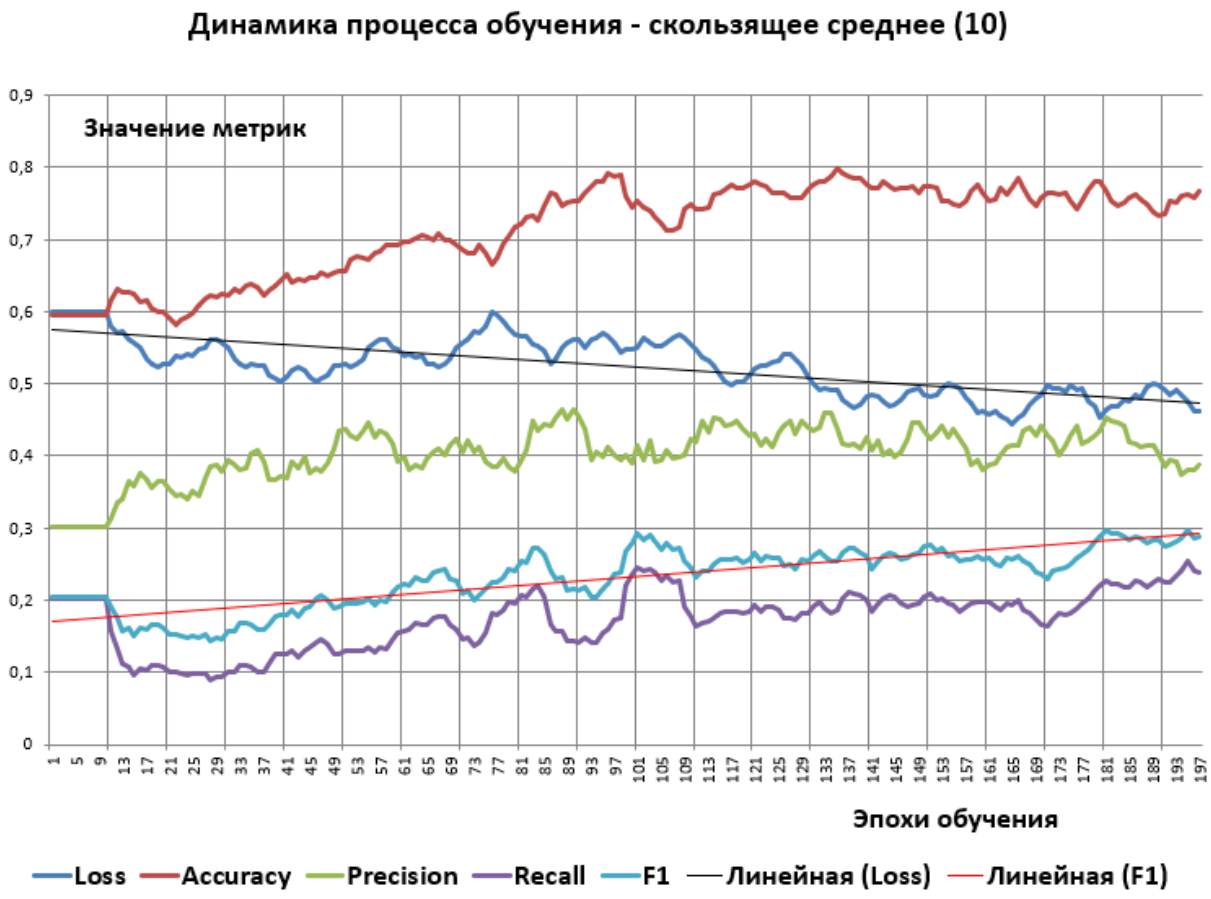


Рисунок 3.5 – График изменения метрик

Код реализации процесса обучения представлен в приложении Д.

Для тестирования модели берется тестовая подвыборка, из нее создаётся окно данных, тек же, как и в процессе обучения. Код для тестирования модели приставлен на рисунке 3.6.

```
for Id, group in all_train_data.groupby('Id'):
    train, val, test = group_split(group)

    individual_window = WindowGenerator(
            input_width=window_input_width, label_width=window_label_width,
            shift=window_shift,
            train_df=train.drop(['Id'], axis=1),
            val_df=val.drop(['Id'], axis=1),
            test_df=test.drop(['Id'], axis=1),
            label_columns=features)

    print("STEP!")
    _, fscore = lstm_model.evaluate(individual_window.test)
    fscorelist.append(fscore)
    print("F1Score: ", fscore)

```

Рисунок 3.6 – Тестирование модели

По результатам тестирования модели получаем среднюю F1-Score равную 0.609 и медиану F1-Score равную 0.669. Accuracy в среднем равную 0.744 и Precision равный 0.589. Полученные результаты незначительно уступают данным, приведенным в работе [9]. С учетом этого в дальнейшем предполагается провести оптимизацию модели нейросети.

### 3.5 Конвертирование модели нейронной сети

При использовании моделей распознавания на основе нейронных сетей для мобильных приложений часто реализуется клиент-серверная архитектура, при этом нейронная сеть реализуется на сервере, т.к. требует больших вычислительных ресурсов.

Поскольку при решении поставленной задачи была получена достаточно компактная нейронная сеть, а приложение не является критическим по времени, было принято решение интегрировать обученную нейронную сеть непосредственно в приложение. Это обеспечит более высокую автономность его работы, позволит проще дообучать модель, при необходимости, а также в общем упростит архитектуру приложения и позволит им пользоваться даже без доступа в интернет.

Конвертирование модели для использования её в мобильном приложении может быть осуществлено средствами библиотеки tensorflow. Так как модель содержит слой LSTM рекомендуется использовать библиотеку tf-nightly. Это экспериментальная библиотека tensorflow.

Устройство конвертированной сети, представленное в графическом виде, с помощью сайта netron [17] приведено на рисунке 3.7.

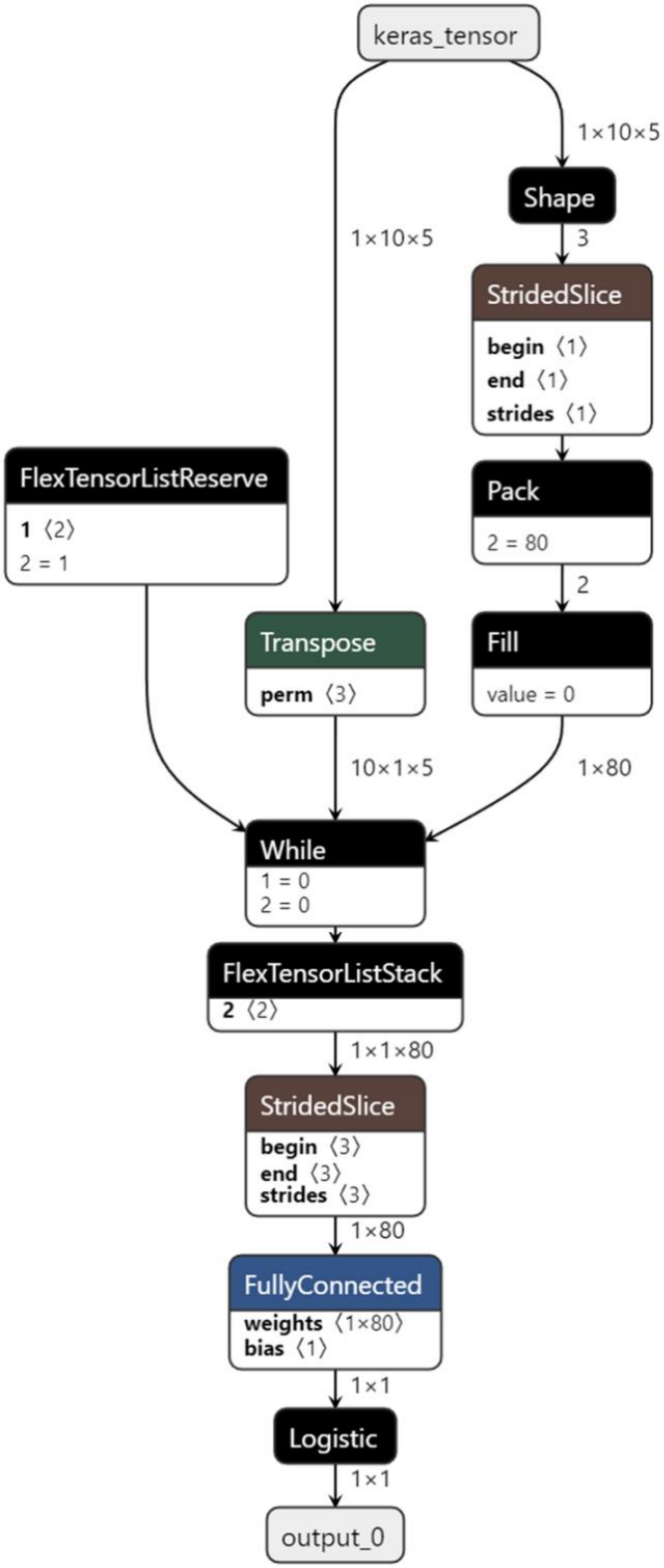


Рисунок 3.7 – Графическое представление архитектуры конвертированной сети

Код, использованный для конвертации модели, представлен на рисунке 3.8.

```
# Конвертируем модель.
run_model = tf.function(lambda x: lstm_model(x))
# ВАЖНО. Поправляем размерности.
BATCH_SIZE = 1
STEPS = 10
INPUT_SIZE = 5
concrete_func = run_model.get_concrete_function(
    tf.TensorSpec([BATCH_SIZE, STEPS, INPUT_SIZE], lstm_model.inputs[0].dtype))

converter = tf.lite.TFLiteConverter.from_keras_model(lstm_model)

converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_ops = [
    tf.lite.OpsSet.TFLITE_BUILTINS,
    tf.lite.OpsSet.SELECT_TF_OPS
]
converter.experimental_new_converter = True

tflite_model = converter.convert()

# Сохраняем конвертированную модель.
with open('test.tflite', 'wb') as f:
    f.write(tflite_model)
```

Рисунок 3.8 – Код конвертирования модели

Эксперименты показали, что время срабатывания нейронной сети после ее портирования на мобильный телефон составляет 1 секунду, что удовлетворяет требованиям к системе.

### 3.6 Разработка мобильного приложения

Для разработки мобильного приложения воспользуемся средой Android Studio. В нём создаём проект с простым графическим приложением. Для сохранения базы данных будем использовать firebase.

#### 3.6.1 Алгоритм работы приложения

Так как нам необходимо, чтобы наша система работала непрерывно, и при этом пользователь взаимодействовал с ним через графический интерфейс, то приложение реализуется через сервис, работающий в фоновом режиме, и не

отображающийся графически, а также через ряд активностей для графического интерфейса.

В графическом интерфейсе пользователь только управляет работой приложения.

Сервис работает в фоновом режиме. Алгоритм работы сервиса после его активации, следующий:

1. выполняется опрос датчика-акселерометра телефона на частоте 10Гц с получением данных с датчика;

2. данные с датчика-акселерометра преобразуются в подходящий для модели формат;

3. окно данных с датчика передаётся модели;

4. если ответ модели не указывает на возникновение эпизода замирания походки, сервис продолжает работу в фоновом режиме; если был обнаружен эпизод замирания походки – сервис инициирует подачу стимулирующего воздействия (вибросигнала) и вызывает графический интерфейс пользователя согласно прецеденту «Обнаружен эпизод», а также сохраняет запись в базу данных.

### 3.6.2 Создание графического интерфейса

В графическом интерфейсе главного экрана пользователь может перейти к настройкам и записям базы данных, а также запустить или отключить сервис. В настройках пользователь может выбрать язык, на котором отображаются надписи в приложении, и тип стимулирующего воздействия из звукового и вибрационного.

В интерфейсе главной страницы присутствует элемент интерфейса, позволяющий включить и выключить приложение. Элемент для перехода к настройкам, элемент для перехода к просмотру базы данных и элемент для того, чтобы закрыть приложение.

Графический интерфейс главной страницы приложения представлен на рисунке 3.9.

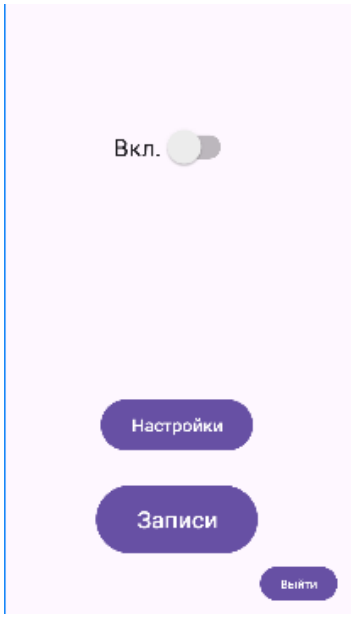


Рисунок 3.9 – Графический интерфейс основного приложения

В настройках присутствуют только элементы позволяющие выбрать тип стимулирующего воздействия и язык интерфейса. Так же присутствует кнопка, для возвращения с текущей страницы.

Графический интерфейс страницы настроек приложения представлен на рисунке 3.10.

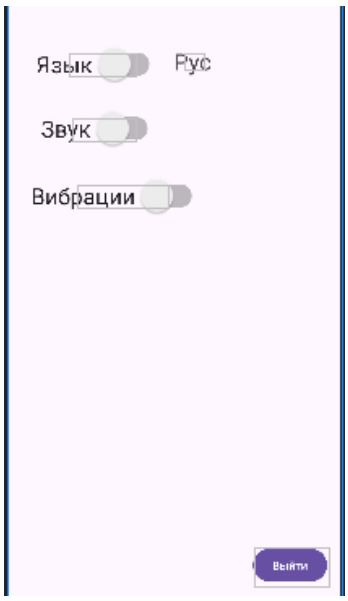


Рисунок 3.10 – Графический интерфейс настроек

В остальных окнах интерфейс примитивен.

В интерфейсе просмотра базы присутствует элемент, позволяющий выводить текст и вернутся с текущей страницы на главную.

В окне графического интерфейса, появляющемся при подаче стимулирующего воздействия, присутствует только кнопка для закрытия окна, по нажатию на которую окно закрывается и приложение продолжает работу в фоновом режиме.

### 3.6.3 Создание сервиса

Так как обработка данные должна происходить непрерывно в фоновом режиме, нам необходимо создать сервис. В нём мы будем получать данные с датчика-акселерометра устройства и передавать их нашей модели.

Получение данных с датчика-акселерометра реализовано с помощью. Встроенных в библиотеку средств обращения с Android API.

Код получения данных с датчика-акселерометра представлен на рисунке 3.11.

```
    val sListener = object : SensorEventListener {
        override fun onSensorChanged(event: SensorEvent?) {
            event?.values?.let { values ->
                // Collect data with a timestamp
                val currentTime = timeStep * 10
                val dataEntry = floatArrayOf(currentTime.toFloat(), values[0], val-
ues[1], values[2])

                accelerometerData.add(dataEntry)
                timeStep++
            }
        }

        override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) { }
    }
```

Рисунок 3.11 – Код получения данных с акселерометра

Для соответствия данных с данными, на которых мы обучали модель устанавливаем частоту опроса датчика в 10Гц, а также создаём временные метки.

Команды для взаимодействия с мобильной версией нейронной сети представлены на рисунке 3.12.

```
val model = TestLstm.newInstance(context)

val inputFeature0 = TensorBuffer.createFixedSize(intArrayOf(1, 10, 5),
DataType.FLOAT32)
inputFeature0.loadBuffer(byteBuffer)

val outputs = model.process(inputFeature0)
val outputFeature0 = outputs.outputFeature0AsTensorBuffer

model.close()
```

Рисунок 3.12 – Команды для взаимодействия с нейронной сетью

Для обработки данных, нам необходимо загрузить ранее обученную модель. Это выполняется с помощью стандартных функций библиотеки tensorflow-lite при запуске сервиса.

Библиотека требует для взаимодействия с моделью, перевести данные в tensorBuffer. Это действие выполняется сервисом при передаче данных модели. Если выход модели содержит положительный результат, мы создаём стимулирующее воздействие.

Код обработки данных представлен на рисунке 3.13.

```
private fun processSensorData() {
    println("Processing data")

   val inputFeature0 = TensorBuffer.createFixedSize(intArrayOf(1, accelerometer-
Data.size, 4), DataType.FLOAT32)

    val buffer = ByteBuffer.allocateDirect(accelerometerData.size * 4 *
Float.SIZE_BYTES)
    val floatBuffer = buffer.asFloatBuffer()
    accelerometerData.forEach { array ->
        floatBuffer.put(array)
    }
    floatBuffer.rewind()  // Очищаем буфер для новых записей

    inputFeature0.loadBuffer(buffer)

    // Передаём данные в модель
     val outputs = lstmModel.process(inputFeature0)
     val outputFeature0 = outputs.outputFeature0AsTensorBuffer

    if (checkForTrigger(outputFeature0)) {
        launchNewActivity()
    }
}
```

Рисунок 3.13 – Код обработки данных моделью

### 3.6.4 Выбор языка приложения и вида стимулирующих воздействий

В настройках мы позволяем пользователю выбрать:

- язык приложения – русский или английский;

- тип стимулирующего воздействия – звуковой или вибрационный.

Для того, чтобы можно было сменить язык, все надписи в приложении задаются через файл strings, который, с помощью редактора, выбранной нами среды, переводится на другие языки. Далее мы устанавливаем локализацию при нажатии кнопки.

Для выбора стимулирующего воздействия, при выборе соответствующих пунктов в интерфейсе настроек, в программе записывается выбранные состояния, которые будут проверяться при попытке подать стимулирующее воздействие.

### 3.6.5 Создание стимулирующего воздействия

Для создания стимулирующего воздействия будем подавать вибрации и воспроизводить стандартное уведомление, выбранное у пользователя в системе.

Что бы воспроизвести только выбранные стимулирующие воздействия воспользуемся сохранёнными значениями, выбранным в настройках.

## 3.7 Выводы по главе 3

На основе ряда экспериментов разработана модель нейронной сети с оптимальной структурой. Проведены эксперименты по обучению модели сети с различным принципом подачи данных. Выбран наиболее удачный из них. Модель обучена, и протестирована. Модель обучена до метрик, практически совпадающих с метриками аналогичных решений. Так же модель успешно конвертирована, для использования в мобильном приложении.

Разработано мобильное приложение, использующее конвертированную модель. Мобильное приложение в полной мере выполняет постеленные нами задачи.

Для поддержки проекта потребуются средства, описанные в главе 2.6. В проекте Android Studio необходимо проследить, чтобы в проекте использовалась библиотека tenerflow-nightly версии 2.18.0-dev20240606 или старше.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы были проанализированы:

– проблема распространения заболевания Паркинсона в мире;

– системы позволяющие выявлять симптом Замирания Походки их сильные и слабые стороны;

Разработана модель нейронной сети, выявляющей эпизод Замирания походки. Создано мобильное приложение, использующее модель для выявления эпизода Замирания Походки, подачи стимулирующего воздействия при его возникновении и записи о произошедших эпизодах.

Для удобства использования приложения его целевой аудитории были разработаны удобные элементы управления.

Исходный код приложения доступен для скачивания с git-репозитория [18].

Результаты представлены на конференции «Проспект Свободный», что подтверждается сертификатом (приложение А), и планируется к опубликованию в сборнике материалов конференции.

Выбрано направление для развития системы для улучшения точности обнаружения эпизодов замирания походки, а также улучшения удобства графического интерфейса.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Breckenridge, C. B. Association between Parkinson’s Disease and Cigarette Smoking, Rural Living, Well-Water Consumption, Farming and Pesticide Use / C. B. Breckenridge, C. Berry, E. T. Chang, R. L. Sielken, J. S. Mandel // Systematic Review and Meta-Analysis: PloS one. — 2016. — Т.11. — №4. — С. e0151841.

2. Перспективы мирового народонаселения // Организация Объединённых Наций: официальный сайт. — 2023. — URL: https://population.un.org/wpp/Download/Standard/Population/ (дата обращения 27.12.2023)

3. Sigcha, L. Deep Learning Approaches for Detecting Freezing of Gait in Parkinson’s Disease Patients through On-Body Acceleration Sensors / L. Sigcha, N. Costa, I. Pavón, S. Costa, et. al. // Sensors. — 2020. — Т.20. — №7. — С. 1895.

4. Murthy, M. Neurodegenerative movement disorders: An epigenetics perspective and promise for the future / M. Murthy, Y. Y. Cheng, J. L. Holton, C. Bettencourt // Neuroradiology and Applied Neurobiology. — 2021. — Т.47. — №7. — С. 897-909.

5. Balestrino, R. Parkinson disease / R. Balestrino, A. Schapira // European Journal of Neurobiology. — 2020. — Т.27. — №1. — С 27-42.

6. Gao, C. Freezing of gait in Parkinson’s disease: pathophysiology, risk factors and treatments / C. Gao, J. Liu, Y. Tan, S. Chen // Translational Neurodegeneration. — 2020. — Т.9. — №12. — С. 1-22.

7. Weiss, D. Freezing of gait: understanding the complexity of an enigmatic phenomenon / D. Weiss, A. Schoellmann, M. D. Fox, N. I. Bohnen, et. al. // Brain a journal of Neurobiology. — 2019. — Т.143. — №1. — С. 14-30.

8. Moore, S. Ambulatory monitoring of freezing of gait in Parkinson’s disease / S. Moore, H. MacDougall, W. Ondo // Journal of Neuroscience Methods. — 2007. — Т.167. — №2. — С. 8-340.

9. Huang, T. Recent trends in wearable device used to detect freezing of gait and falls in people with Parkinson’s disease: A systematic review / T. Huang,

M. Li, J. Huang // Parkinson’s Disease and Aging-related Movement Disorders. — 2023. — Т.15. — №15. — С. 1119956.

10. Mazilu, S. Online Detection of Freezing of Gait with Smartphones and Machine Learning Techniques / S. Mazilu, M. Hardegger, Z. Zhu, D. Roggen, G. Troster, et. al. // International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops. — 2012. — Т.6. — №2. — С. 248680.

11. Borz, L. Detection of freezing of gait in people with Parkinson’s disease using smartphones / L. Borz, G. Olmo, C. A. Artusi, L. Lopiano // Annual Computers, Software, and Applications Conference. — 2020. — Т.44 — №1. — С. 0-186.

12. Ahmad, J. Design and Implementation of an Instrumented walking cane for Detection of Freezing of Gait / J. Ahmad, A. El-Asmar, R. A. Daou, A. Hayek, J. Boercsoek // International Multidisciplinary Conference on Engineering Technology. — 2021. — Т.3. — №1. — С. 08-10.

13. Masiala, S. Feature-Set-Engineering for Detecting Freezing of Gait in Parkinson's Disease using Deep Recurrent Neural Networks / S. Masiala, W. Huijbers, M. Atzmueller // Eindhoven University of Technology. — 2019. — Т. 2019. — №1. — С. 1909.03428.

14. Demrozi, F. Towards a wearable system for predicting freezing of gait in people affected by Parkinson’s disease / F. Demrozi, R. Bacchin, S. Tamburin, M. Cristani, G. Pravadelli // Journal of Biomedical and Health Informatics. — 2019. — Т. 24. — №9. — С. 2168-2194.

15. Рекуррентные нейронные сети // Git-Hub: сайт. — 2023. — URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения дата обращения 27.11.2023)

16. Выявление замирания походки // Kaggle: сайт. — 2023. — URL: https://www.kaggle.com/competitions/tlvmc-parkinsons-freezing-gait-prediction/code (дата обращения 27.12.2023)

17. Визуализация моделей // Neutron: сайт. — 2024. — URL: https://netron.app/ (дата обращения 03.05.2024)

18. Репозиторий разработанного приложения // Git-Hub: сайт. — 2024. — URL: https://github.com/DrWhilson/FOG-Decection-with-Phone (дата обращения 17.05.2024)


СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ | SIBERIAN FEDERAL UNIVERSITY

ПРОСПЕКТ СВОБОДНЫЙ

# СЕРТИФИКАТ

подтверждает, что

## Липатов Евгений Андреевич

принял(-а) **заочное участие** в юбилейной XX Международной научной конференции студентов, аспирантов и молодых ученых «ПРОСПЕКТ СВОБОДНЫЙ – 2024»

Руководитель направления
по молодежной науке
Офиса развития научной
деятельности СФУ

К. А. Кистерский

№ 33777 -2024

# ПРИЛОЖЕНИЕ Б

## Код загрузки базы данных

```
def get_train_data(targets, features, accmeasurs):
    # !Пути к папкам
    path = r'..\DATA'
    train_path = 'train'

    # !Пути к файлам
    defog_path = 'defog'
    tdcsfog_path = 'tdcsfog'
    sample = pd.read_csv(os.path.join(path, 'sample_submission.csv'))
    sample.head()
    sample.info()

    # !Загружаем данные
    # Загружаем tdcsfog_path
    tdcsfog_df = get_tdcsfog_full(path, train_path, tdcsfog_path)

    # Загружаем defog_path
    defog_df = get_defog_full(path, train_path, defog_path)

    # Объеденяем данные
    all_train_data = pd.concat([tdcsfog_df, defog_df])
    all_train_data = all_train_data.astype({'Time': 'int32', 'Turn': 'int8', 'Walk-
ing': 'int8',
                                            'StartHesitation': 'int8', 'AccV':
'float16',
                                            'AccML': 'float16', 'AccAP':
'float16'})
    defog_df = None
    tdcsfog_df = None

    # Удаляем аномалии
    all_train_data = all_train_data.loc[(all_train_data[['AccV', 'AccML', 'AccAP']]
<= 9.81).all(axis=1)]

    # Объеденяем типы событий
    all_train_data['Event'] = (
                all_train_data['Turn'] | all_train_data['Walking'] |
all_train_data['StartHesitation']).astype(int)

    # Удаляем старые типы событий
    all_train_data = all_train_data.drop('Turn', axis=1)
    all_train_data = all_train_data.drop('Walking', axis=1)
    all_train_data = all_train_data.drop('StartHesitation', axis=1)

    # Сжимаем таблицу по времени
    all_train_data = all_train_data[::10]

    # !Create All Feature
    all_features = [feature for feature in all_train_data.columns if
                    feature != 'Id' and feature not in targets and feature !=
'Time']
    print(all_features)

    return all_features, all_train_data
```

# ПРИЛОЖЕНИЕ В

## Код класса генератора окон

```
class WindowGenerator:
    def __init__(self, input_width, label_width, shift,
                 train_df, val_df, test_df,
                 label_columns=None):
        # Сохраняем сырые данные
        self.train_df = train_df
        self.val_df = val_df
        self.test_df = test_df

        # Создаём метки для лэйблов
        self.label_columns = label_columns
        if label_columns is not None:
            self.label_columns_indices = {name: i for i, name in
                                          enumerate(label_columns)}
        self.column_indices = {name: i for i, name in
                               enumerate(train_df.columns)}

        # Создаём необходимые параметры
        self.input_width = input_width
        self.label_width = label_width
        self.shift = shift

        self.total_window_size = input_width + shift

        self.input_slice = slice(0, input_width)
        self.input_indices = np.arange(self.total_window_size)[self.input_slice]

        self.label_start = self.total_window_size - self.label_width
        self.labels_slice = slice(self.label_start, None)
        self.label_indices = np.arange(self.total_window_size)[self.labels_slice]
```

# ПРИЛОЖЕНИЕ Г

## Реализация F1-Score метрики

```
 1  def F1_score(y_true, y_pred):
 2      def recall(y_true, y_pred):
 3          true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
 4          possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
 5          recall = true_positives / (possible_positives + K.epsilon())
 6          return recall
 7
 8      def precision(y_true, y_pred):
 9          true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
10          predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
11          precision = true_positives / (predicted_positives + K.epsilon())
12          return precision
13
14      precision = precision(y_true, y_pred)
15      recall = recall(y_true, y_pred)
16      return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

# ПРИЛОЖЕНИЕ Д

## Код обучения модели

```
lookback = 3
targets = ['Event']
features = ['Time', 'AccV', 'AccML', 'AccAP']
acc_measures = ['AccV', 'AccML', 'AccAP']
all_features, all_train_data = get_train_data(targets, features, acc_measures)
# Получаем данные первого пациента
ids = all_train_data['Id'].unique()
characteristic_group = all_train_data[all_train_data['Id'] == ids[0]]

# Разделяем данные на тренеровачные, тестовые и валидационные
train, val, test = group_split(characteristic_group)

# Создаём константы
window_input_width = 10
window_label_width = 1
window_shift = 0
epochs = 20
losses = ['binary_crossentropy']
metrics = [F1_score]

# Создаём окно данных
characteristic_window = WindowGenerator(
    input_width=window_input_width, label_width=window_label_width,
shift=window_shift,
    train_df=train.drop(['Id'], axis=1),
    val_df=val.drop(['Id'], axis=1),
    test_df=test.drop(['Id'], axis=1),
    label_columns=features)

# Создаём модель
lstm_model = LSTMModel(characteristic_window, features, lookback)
lstm_model.model.compile(loss=losses, optimizer=tf.keras.optimizers.Adam(), met-
rics=metrics)
lstm_model.model.summary()

# Обучаем модель
for Id, group in all_train_data.groupby('Id'):
    # Разделяем данные на тренеровачные, тестовые и валидационные
    train, val, test = group_split(group)

    print("!Len: ", len(train))
    print("!Events:", train['Event'].value_counts())

    # Создаём окно данных
    individual_window = WindowGenerator(
        input_width=window_input_width, label_width=window_label_width,
shift=window_shift,
        train_df=train.drop(['Id'], axis=1),
        val_df=val.drop(['Id'], axis=1),
        test_df=test.drop(['Id'], axis=1),
        label_columns=features)
    lstm_model.model.fit(individual_window.train, epochs=epochs,
                         validation_data=individual_window.val)
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
«17» 06 2024 г.

# БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Система выявления симптома замирания походки
у пациентов с болезнью Паркинсона

| | | | | |
|---|---|---|---|---|
| Руководитель | 17.06.24 | доцент, канд. техн. наук | Н.Ю. Сиротинина | |
| | подпись дата | должность, ученая степень | инициалы, фамилия | |
| Выпускник | 17.06.24 | | Е.А. Липатов | |
| | подпись дата | | инициалы, фамилия | |
| Нормоконтролер | 17.06.24 | доцент, канд. техн. наук | Н.Ю. Сиротинина | |
| | подпись дата | должность, ученая степень | инициалы, фамилия | |

Красноярск 2024