

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА»

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
КАФЕДРА ПРИКЛАДНОЙ МЕХАНИКИ И УПРАВЛЕНИЯ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(ДИПЛОМНАЯ РАБОТА)
специалиста

**НАВИГАЦИЯ ПЕШЕХОДА ПО
ИНЕРЦИАЛЬНЫМ ДАТЧИКАМ
СМАРТФОНА МЕТОДАМИ
МАШИННОГО ОБУЧЕНИЯ.**

Выполнил студент
621 группы
Никитин В. С.

подпись студента

Научный руководитель:
д.ф.-м.н., профессор Болотин Ю. В.

подпись научного руководителя

Москва
2022

Содержание

1 Введение	2
2 Обозначения, используемые в работе	2
3 Постановка задачи	2
4 Обзор литературы	4
4.1 Методы без использования нейронных сетей	4
4.2 Методы с использованием нейронных сетей	5
5 Недостаток использования метода инерциальной навигации при решении данной задачи	5
6 Описание нейронных сетей	5
6.1 Искусственный нейрон	5
6.2 Искусственная нейросеть	6
6.3 Обучение нейронной сети	7
6.4 Рекурентные нейронные сети	8
7 Сети LSTM	10
7.1 Архитектура LSTM	10
7.2 Входной блок	12
7.3 Блок забывания	13
7.4 Выходной блок	14
7.5 Борьба с затухающим градиентом	14
8 Методика определения изменения направления ходьбы и пройденного расстояния	16
8.1 Исходные данные	16
8.2 Описание используемых устройств	18
8.3 Синхронизация данных	18
8.4 Подготовка данных для обучения	18
8.5 Архитектура используемой в работе нейронной сети	20
8.6 Результаты	20
9 Навигация с использованием комбинации фильтра Мэджвика и метода главных компонент	24
9.1 Описание метода	24
9.2 Результаты	26
9.3 Недостатки метода	27
9.4 Сравнение методов	28
10 Выводы	29
Список используемой литературы	30

1 Введение

Рассматривается задача навигации пешехода с использованием смартфона. В дипломной работе предполагается решить поставленную задачу, ограничившись внутренними средствами смартфона — акселерометрами и датчиками угловой скорости, а так же проанализировать варианты навигации с использованием магнитометра и без него, так как часть данных получена в помещениях, где показания магнитометров сильно зашумлены. Алгоритм будет строиться с использованием нейронных сетей. Также стоит отметить, что использованный в дипломной работе алгоритм работает в реальном времени, так как на каждом шаге для предсказания направления движения и пройденного расстояния использует информацию лишь за 2 секунды. Для обучения нейронной сети будет использоваться набор имеющихся экспериментальных данных, а также данные из открытых источников. Произведено сравнение с результатами курсовых прошлых лет. Для проверки точности навигации используется информация, полученная с IMU, установленной на стопе.

Также рассмотрен алгоритм, где ориентация смартфона определяется с использованием фильтра Мэджвика [12] по показаниям акселерометра, ДУС и магнитометра, а траектория строится методом главных компонент.

2 Обозначения, используемые в работе

Показания акселерометра в момент времени t_k обозначим $a_k = (a_{k_x}, a_{k_y}, a_{k_z})$. Показания ДУС в момент времени t_k обозначим $\omega_k = (\omega_{k_1}, \omega_{k_2}, \omega_{k_3})$.

l — пройденное пешеходом расстояние. $\Delta\psi$ — изменение направления движения.

W_i — матрица весов. W_0 — смещение нейрона.

F — функция активации.

J — функция потерь.

η — скорость обучения нейронной сети (это параметр градиентных алгоритмов обучения нейронных сетей, позволяющий управлять величиной коррекции весов на каждой итерации).

RNN (recurrent neural network) — рекурентная нейронная сеть.

$LSTM$ (long short-term memory) — сеть долгой краткосрочной памяти.

x_t — вход нейронной сети в момент времени t в случае сетей работающими с временными рядами. В противном случае, x_i — i -ый вход нейронной сети

y_t — выход нейронной сети в момент времени t в случае сетей работающими с временными рядами. В противном случае, y_i — i -ый выход нейронной сети.

h_t — скрытое состояние, полученное в момент времени t в случае сетей работающими с временными рядами. В противном случае, h_i — i -ый элемент скрытого слоя.

C_t — состояние ячейки LSTM, полученное в момент времени t .

$tanh$ — гиперболический тангенс. σ — сигмовидная функция.

3 Постановка задачи

Наша задача состоит в том, чтобы построить траекторию пешехода по информации, полученной со смартфона. Исходными данными являются показания акселерометров ($a_k = (a_{k_x}, a_{k_y}, a_{k_z})$, рис. 1) и ДУС ($\omega_k = (\omega_{k_x}, \omega_{k_y}, \omega_{k_z})$, рис. 2), записанные в моменты времени $t_k = k\Delta t$ с частотой 200 Гц ($\Delta t = 0.005c$) и 400 Гц ($\Delta t = 0.0025c$).

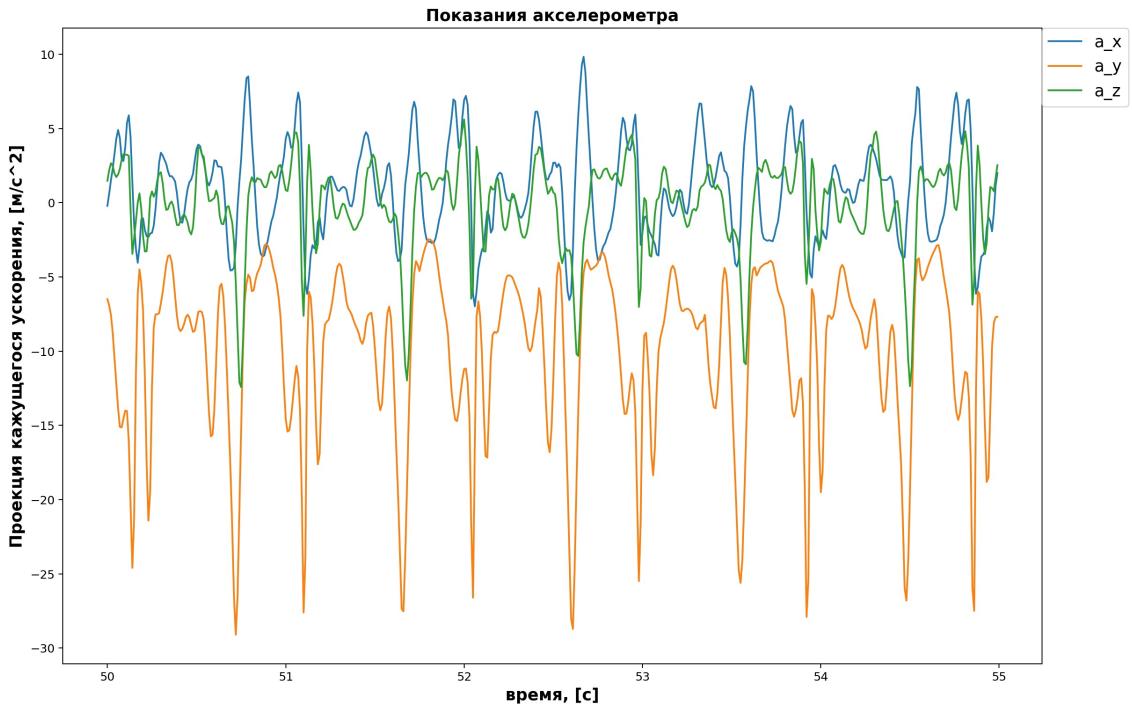


Рис. 1: Показания акселерометра смартфона.

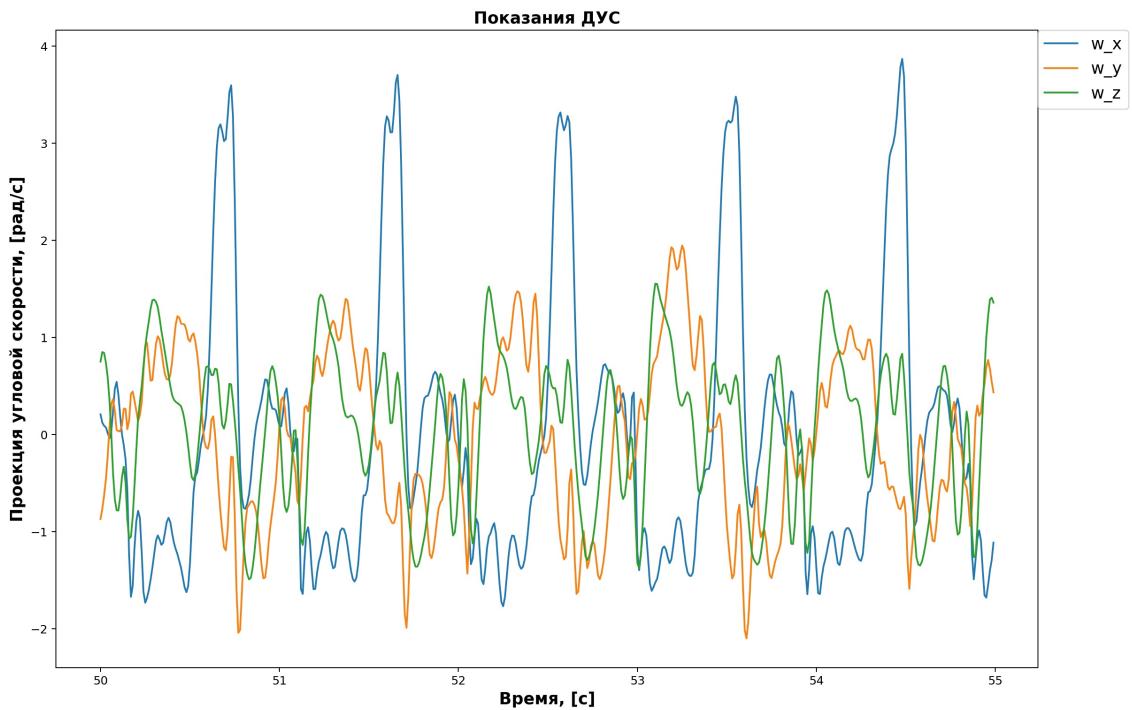


Рис. 2: Показания ДУС смартфона.

В качестве истинных траекторий используются траектории, полученные с IMU, расположенные на стопах пешехода, путем решения навигационной задачи [7] (рис. 3).

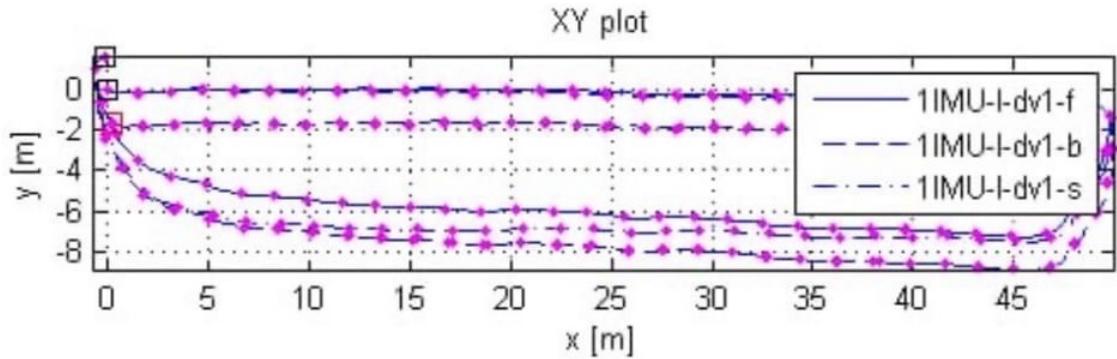


Рис. 3: Оценки траектории левой и правой ноги

Так же будут использоваться показания смартфона и траектории из открытых источников (датасет OxIOD [3]).

Задача будет решаться с помощью нейронных сетей. Для обучения будет сети использоваться показания смартфона и истинные траектории из набора имеющихся данных и из открытых источников. При тестировании сети используются только показания смартфона.

4 Обзор литературы

4.1 Методы без использования нейронных сетей

Бесплатформенные инерциальные навигационные системы и метод на основе интегрирования модельных уравнений были подробно изучены в работе [13]. Инерциальные системы в данной работе основываются на использовании дорогих, тяжелых и высокоточных инерциальных измерительных устройствах, поэтому их основное применение ограничивается движущимися транспортными средствами, такими как автомобили, корабли, самолеты, подводные лодки и космические корабли. Недавние достижения в области технологии MEMS позволяют использовать недорогие MEMS IMU на робототехнике, БПЛА [14] и мобильных устройствах [15]. Однако из-за ограничений по размеру и стоимости точность таких MEMS IMU ограничена и для повышения точности необходимо использование дополнительных датчиков, такими как датчик визуальной инерциальной одометрии [16]. Другим решением является прикрепление IMU к ноге пользователя для использования формализованной информацией о фазе опоры для коррекции накапливающихся ошибок.

Еще одним методом является метод PDR: в отличие от метода, основанного на интегрировании модельных уравнений, PDR использует измерения БИНС для обнаружения шагов, оценки длины шага и направления с помощью эмпирической формулы [17]. Системные ошибки по-прежнему быстро накапливаются из-за неточного определения шага. Кроме того, большое количество параметров необходимо настроить в соответствии с особенностями ходьбы пешехода. Недавние исследования в основном были сосредоточены на объединении PDR с внешними внешними источниками информации, такими как план этажа [18] или использовании информации WiFi [19], по-прежнему оставляя ряд нерешенных проблем связанных с дрейфом датчиков.

4.2 Методы с использованием нейронных сетей

В последнее время нейронные сети показали свою эффективность и преимущество над изложенными ранее методами. Однако, в большинстве случаев помимо показаний акселерометра и ДУС смартфона используют визуальную одометрию [20] и визуальную инерциальную одометрию [21]. Существует сеть RoNIN [22], которая решает задачу автономной навигации пешехода исключительно по инерциальным показаниям смартфона, однако для корректной работы алгоритма необходима начальная выставка для точного определения ориентации смартфона. В работе будет рассмотрена сеть, которая ограничивается лишь использованием акселерометров и ДУС смартфона, а также не нуждающаяся в начальной выставке.

5 Недостаток использования метода инерциальной навигации при решении данной задачи

Вычисление траектории перемещения устройства с акселерометром и ДУС возможно путем интегрирования модельных уравнений [4] (необходимо лишь сделать начальную выставку для определения начальных условий). Однако, в случае IMU на стопах мы обладаем формализованной информацией о фазе опоры (скорость в фазе опоры равна 0), что помогает повысить точность результата, поскольку данные зашумлены. В случае со смартфоном у нас имеется лишь неформализованная информация о характере движения, которую мы должны использовать. В связи с этим фактом в дипломной работе используется метод основанный на использовании нейронных сетей и метод главных компонент.

6 Описание нейронных сетей

6.1 Искусственный нейрон

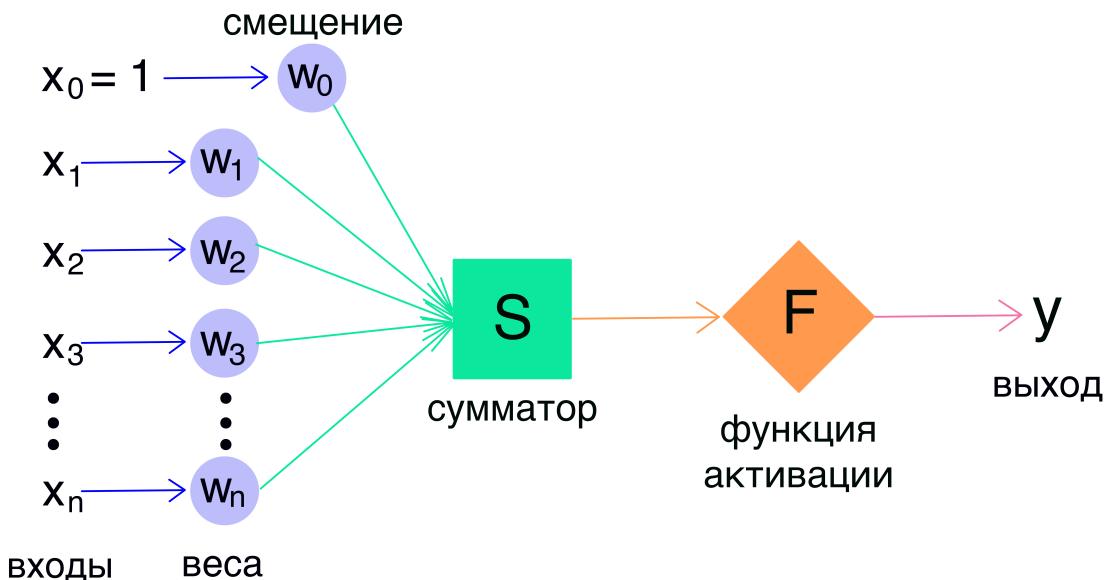


Рис. 4: Искусственный нейрон

Рассмотрим один нейрон [1]. Математически нейрон представляет собой взвешенный сумматор, единственный выход которого определяется через его входы и матрицы весов следующим образом:

$$y = F \left(\sum_{i=1}^n W_i \cdot x_i + W_0 \cdot x_0 \right)$$

Здесь x_i и W_i — соответственно входные сигналы нейрона и матрицы весов. Как правило x_0 берут равным 1 и W_0 называют смещением нейрона (вес единичного входа). Тогда $y = F(\sum_{i=1}^n W_i x_i + W_0)$. Функция F (обычно нелинейная) называется активационной функцией и может иметь различный вид, в том числе приведенные на рисунке 5: гиперболический тангенс (TANH), сигмоида (σ), RELU.

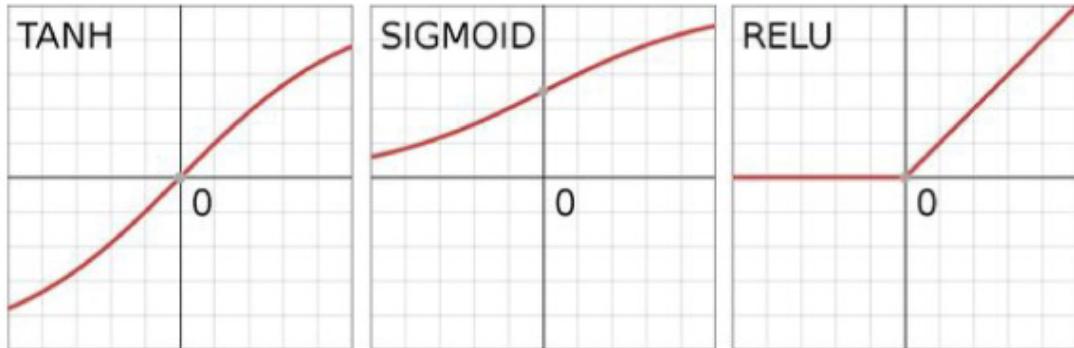


Рис. 5: Активационные функции. Гиперболический тангенс, SIGMOID, RELU.

6.2 Искусственная нейросеть

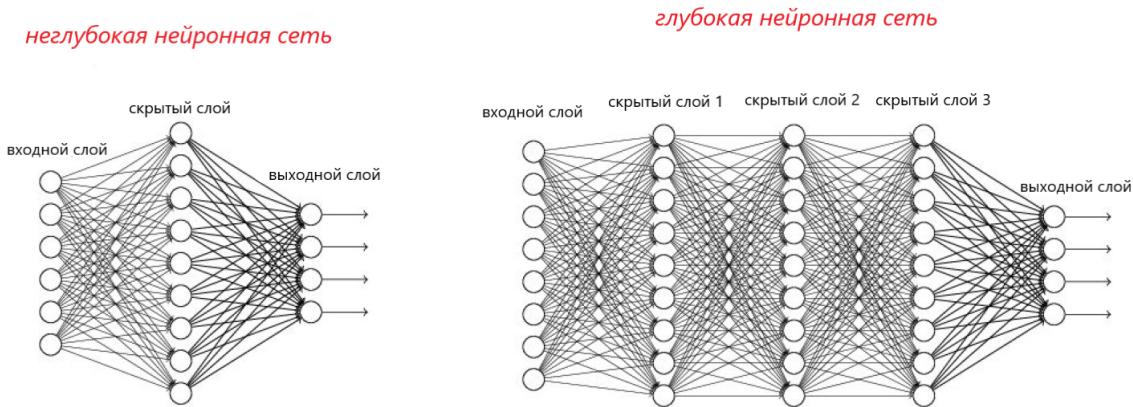


Рис. 6: Пример неглубокой (слева) и глубокой (справа) нейронной сети

Искусственная нейросеть [2] — это способ собрать нейроны в сеть, чтобы она решала определенную задачу. Нейроны собираются по слоям. Есть входной слой, куда подаются входные сигналы, есть выходной слой, откуда снимается результат работы нейросети (выходные сигналы), и между ними есть скрытые слои, их может быть 1,

2, 3, и более. Если скрытых слоев больше, чем 1, нейросеть считается глубокой, если 1, то неглубокой.

Существует огромное разнообразие различных архитектур нейронных сетей.

6.3 Обучение нейронной сети

Обучение нейросетей происходит в два этапа:

1. Прямое распространение ошибки.
2. Обратное распространение ошибки [8].

Во время прямого распространения ошибки делается предсказание ответа. При обратном распространении ошибка между фактическим ответом и предсказанным минимизируется.

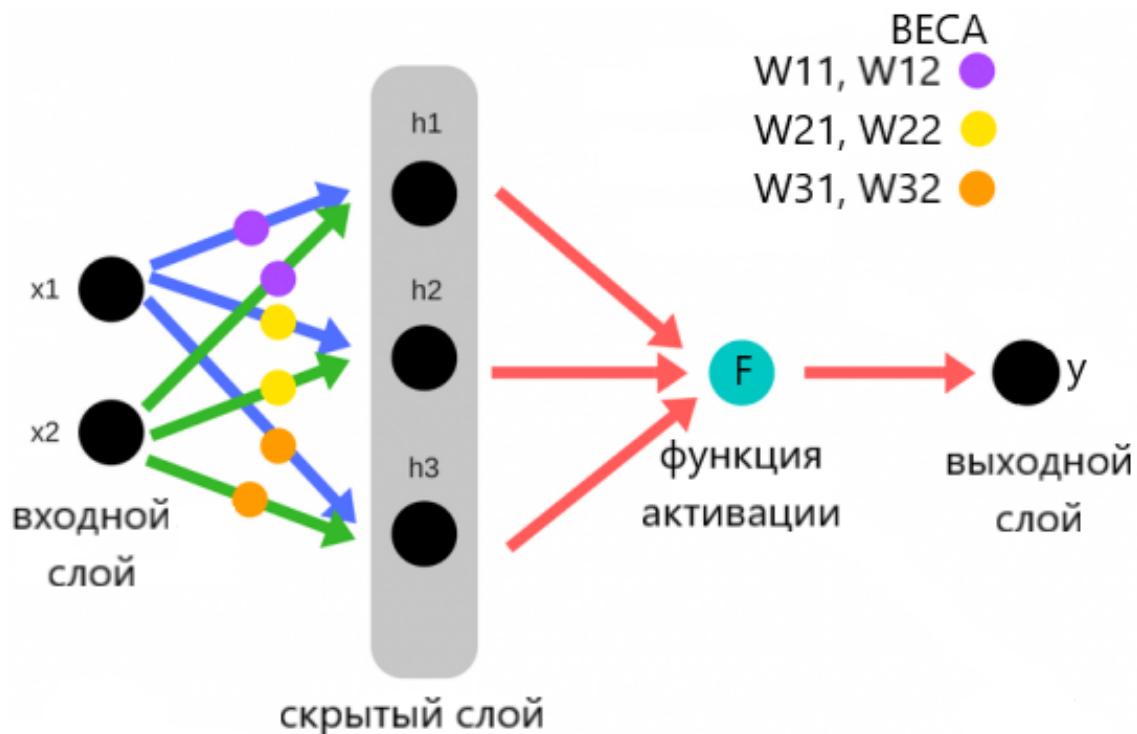


Рис. 7: Прямое распространение ошибки

Зададим начальные матрицы весов случайным образом ($W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, W_{32}$). Умножим входные сигналы на весовые матрицы для формирования скрытого слоя:

$$h_1 = (W_{11} \cdot x_1) + (W_{12} \cdot x_2), h_2 = (W_{21} \cdot x_1) + (W_{22} \cdot x_2), h_3 = (W_{31} \cdot x_1) + (W_{32} \cdot x_2)$$

Выходные данные из скрытого слоя передается через нелинейную функцию (функцию активации), для получения выхода сети:

$$y_1 = F(h_1 + h_2 + h_3)$$

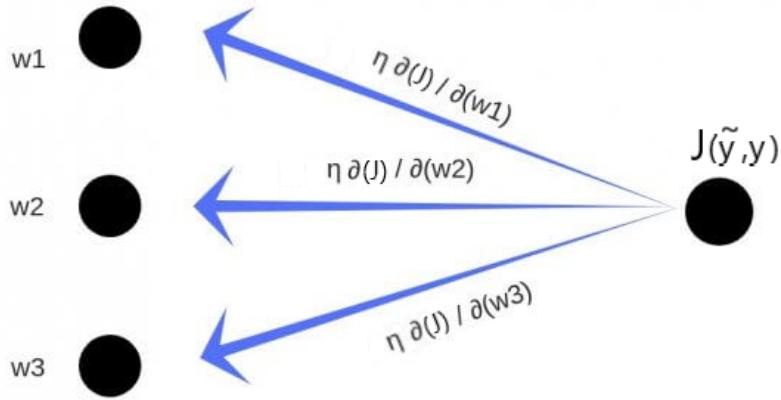


Рис. 8: Обратное распространение ошибки

Суммарная ошибка вычисляется как разность между ожидаемым значением \tilde{y} (из обучающего набора) и полученным значением y (посчитанного на этапе прямого распространения ошибки), проходящих через функцию потерь (J). Частная производная ошибки вычисляется по каждой матрице весов (эти частные производные отражают вклад каждой весовой матрицы в общую ошибку). Затем эти дифференциалы умножаются на число, называемое скоростью обучения (η). Полученный результат затем вычитается из соответствующих весовых матриц.

В результате получатся следующие обновленные весовые матрицы:

$$W_1 \leftarrow W_1 - (\eta \frac{\partial J}{\partial W_1}), W_2 \leftarrow W_2 - (\eta \frac{\partial J}{\partial W_2}), W_3 \leftarrow W_3 - (\eta \frac{\partial J}{\partial W_3})$$

6.4 Рекуррентные нейронные сети

Рекуррентные нейронные сети [6] — сети с циклами, которые хорошо подходят для обработки последовательностей. Идея RNN заключается в последовательном использовании информации. В традиционных нейронных сетях подразумевается, что все входы и выходы независимы, но для многих задач это не подходит. Еще одна интерпретация RNN: это сети, у которых есть «память», которая учитывает предшествующую информацию.

RNN можно развернуть в полную сеть. Разверткой мы просто выписываем сеть для получения полной последовательности.

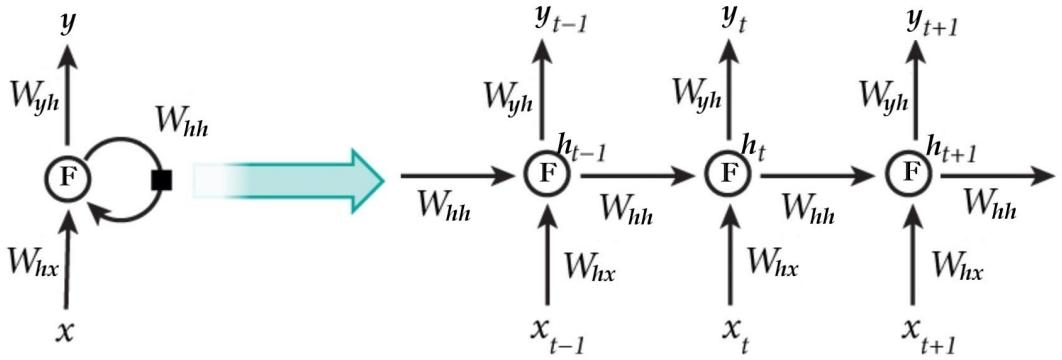


Рис. 9: x_t, y_t - элементы входной и выходной последовательности нейронной сети в момент времени t . h_t - это скрытое состояние, полученное в момент времени t . W_{hx}, W_{hh}, W_{yh} — матрицы весов. Кружком обозначена ячейка RNN. F — функция активации.

Рассмотрим основные вычисления, которые должны выполнять ячейки RNN для создания скрытых состояний и выходных данных:

$$h_t = F(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t)$$

На первом временном шаге (в момент $t = 1$) скрытое состояние h_0 обычно берется нулевой матрицей, чтобы его можно было передать в ячейку RNN вместе с первым элементом входной последовательности. В классической архитектуре RNN для получения скрытого состояния h_t в момент времени t скрытое состояние предыдущей ячейки h_{t-1} и элемент входной последовательности x_t в момент t умножаются на матрицы весов (W_{hh} и W_{hx} соответственно) и суммируются. полученный результат пропускается через функцию активации (как правило берется гиперболический тангенс).

Для получения элемента выходной последовательности y_t в момент времени t необходимо умножить текущее скрытое состояние h_t на соответствующую матрицу весов W_{yh} :

$$y_t = W_{yh} \cdot h_t$$

Скрытое состояние h_t , которое мы только что получили будет передано в следующую ячейку RNN вместе со следующим элементом входной последовательности x_{t+1} , и этот процесс будет продолжаться до тех пор, пока у нас не закончатся входные данные или модель не будет запрограммирована на прекращение создания выходных данных.

RNN имеет одинаковые матрицы весов W_{hx}, W_{hh}, W_{yh} на всех этапах. Это отражает тот факт, что мы выполняем одну и ту же задачу на каждом шаге, используя только разные входы. Это значительно уменьшает общее количество параметров, которые необходимо подбирать.

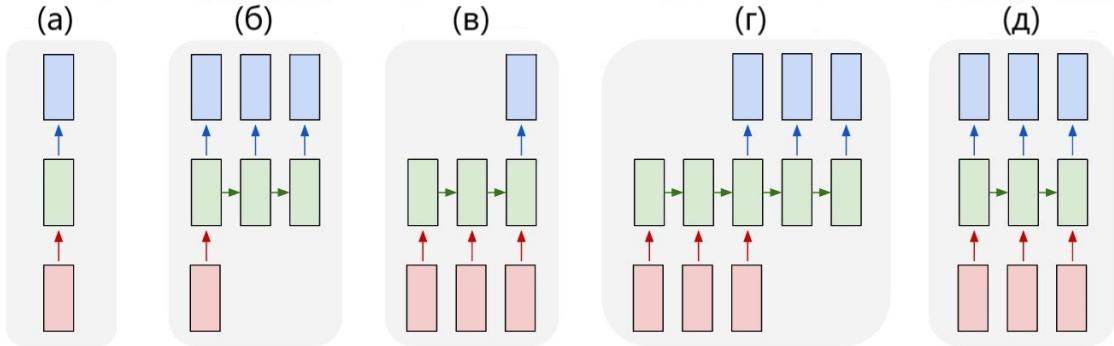


Рис. 10: Виды рекуррентных сетей. Красным цветом обозначены входы, зеленым - ячейки RNN, синим - выходы. (а) - обычная нейронная сеть; (б) - один вход ко многим выходам; (в) - много входов и один выход; (г),(д) - много входов и много выходов

В зависимости от рассматриваемой задачи нам могут понадобиться не все выходы. В дипломной работе используется подход "несколько входов - один выход". По последовательности показаний смартфона (акселерометры и ДУС) мы будем предсказывать двумерный вектор, первая компонента которого является пройденным расстоянием, а вторая - изменением направления движения. Длина последовательности равна 200, т.е. наша сеть будет иметь в качестве входа последовательность длины 200 (каждый элемент последовательность представляет собой шестимерный вектор $x_t = (a_{t_x}, a_{t_y}, a_{t_z}, \omega_{t_x}, \omega_{t_y}, \omega_{t_z})$ - показания акселерометров и ДУС в момент времени t) и всего 1 выход. Выход берется с самой последней ячейки RNN.

Обучение RNN аналогично обучению обычной нейронной сети. Мы также используем алгоритм обратного распространения ошибки (backpropagation), но с небольшим изменением. Поскольку одни и те же параметры используются на всех временных этапах в сети, градиент на каждом выходе зависит не только от расчетов текущего шага, но и от предыдущих временных шагов. Например, чтобы вычислить градиент при $t = 4$, нам нужно было бы «распространить ошибку» на 3 шага и суммировать градиенты. Этот алгоритм называется «алгоритмом обратного распространения ошибки сквозь время» [9] (Backpropagation Through Time, BPTT). Поэтому рекуррентные нейронные сети, прошедшие обучение с BPTT, испытывают трудности с изучением долгосрочных зависимостей из-за затухания/взрыва градиента. Чтобы обойти эти проблемы были разработаны специальные архитектуры RNN (например, LSTM - сеть долгой краткосрочной памяти).

7 Сети LSTM

7.1 Архитектура LSTM

LSTM (long short-term memory) [5] — тип рекуррентной нейронной сети, способный обучаться долгосрочным зависимостям.

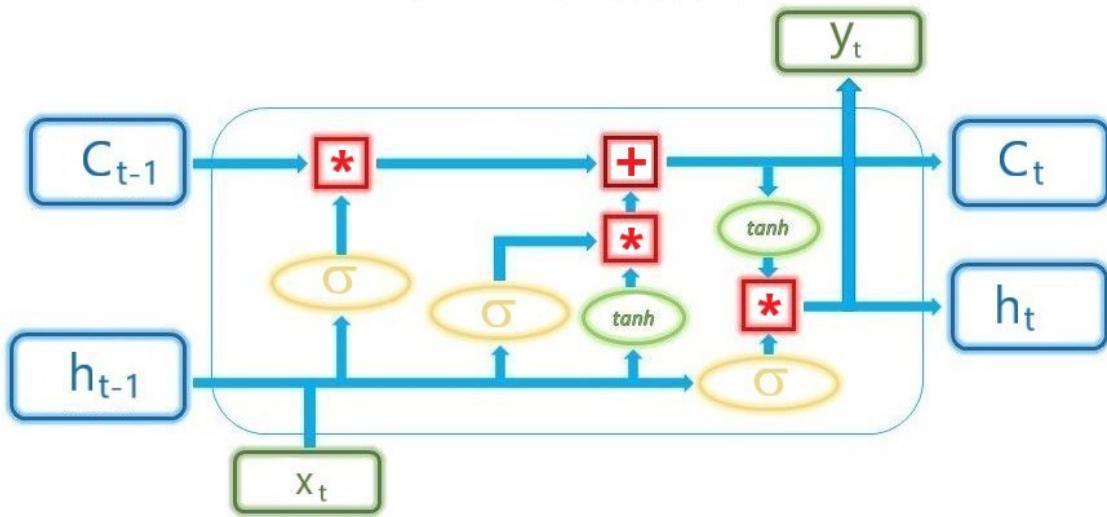


Рис. 11: Архитектура ячейки LSTM. Звездочкой обозначено поэлементное умножение векторов. Знаком "плюс" поэлементная сумма векторов. Сходящиеся стрелки - соединение матриц (конкатенация), расходящиеся стрелки - копирование, $tanh$ - гиперболический тангенс, σ - сигмоидный слой, C_t - состояние ячейки, h_t - скрытое состояние ячейки.

На каждом временном шаге ячейка LSTM получает 3 разных фрагмента информации: текущие входные данные (текущий элемент входной последовательности), кратковременную память из предыдущей ячейки (аналогично скрытым состояниям в RNN) и, наконец, долговременную память. Кратковременную память обычно называют скрытым состоянием, а долговременную память обычно называют состоянием ячейки. Затем в ячейки происходит фильтрация полученной информации, которая должна быть сохранена или отброшена на каждом временном шаге (часть информации сохраняется, часть - отбрасывается). Далее отфильтрованная информация передается следующей ячейки (передаем состояние и скрытое состояние ячейки).

Рассмотрим как в ячейке LSTM происходит фильтрация информации подробнее.

7.2 Входной блок

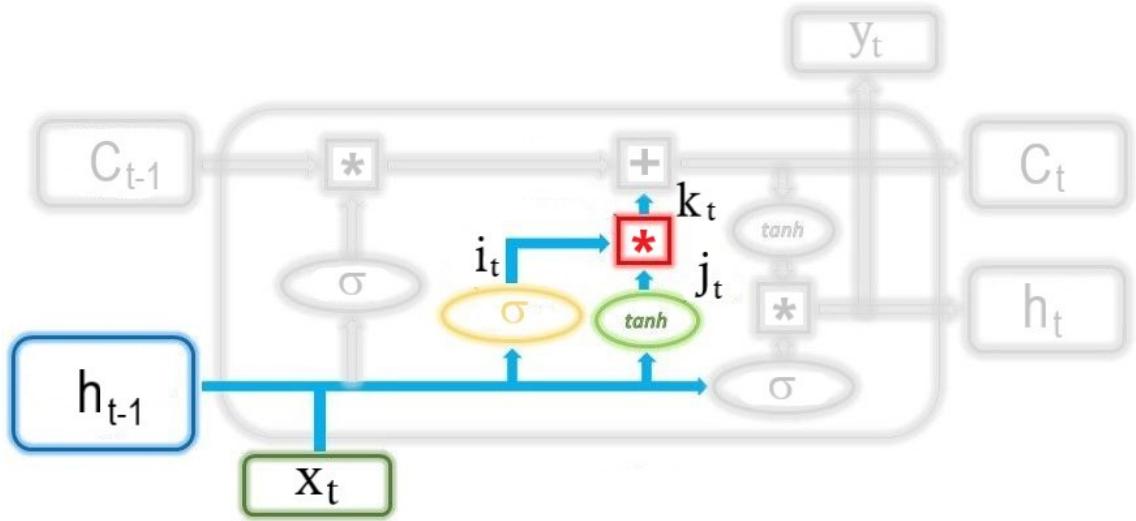


Рис. 12: Входной блок ячейки LSTM

Во входном блоке решается какая новая информация будет храниться в долговременной памяти. Он работает только с текущим элементом входной последовательности и скрытым состоянием предыдущей ячейки (с предыдущего временного шага). Избавление от лишней информации происходит с помощью 2 слоев. В первом слое в качестве функции активации выбирается сигмоида, в которую мы передаем скрытое состоянием предыдущей ячейки и текущий элемент входной последовательности. Сигмовидная функция преобразовывает полученные значения в диапазон от 0 до 1, где 0 указывает, что информации не важна, а 1, что информация будет использоваться. Это помогает определить, какие значения следует сохранить и использовать, а какие следует отбросить.

$$i_t = \sigma(W_i \cdot (h_{t-1}, x_t) + W_{i_0}), \text{ где } W_{i_0} \text{ — смещение}$$

Второй слой также берет скрытое состояние предыдущей ячейки и текущий элемент входной последовательности и пропускает их через функцию активации (выбирается $tanh$, так как его область определения $[-1, 1]$. Отрицательные значения необходимы, если мы хотим уменьшить влияние определенного элемента вектора на состояние ячейки).

$$j_t = \tanh(W_j \cdot (h_{t-1}, x_t) + W_{j_0})$$

По мере обучения (с помощью метода обратного распространения ошибки) обе матрицы весов обновляются таким образом, чтобы через слои пропускалась только полезная информация.

Затем выходные данные этих двух слоев умножаются, и конечный результат представляет собой информацию, которая должна храниться в долговременной памяти.

$$k_t = i_t * j_t$$

7.3 Блок забывания

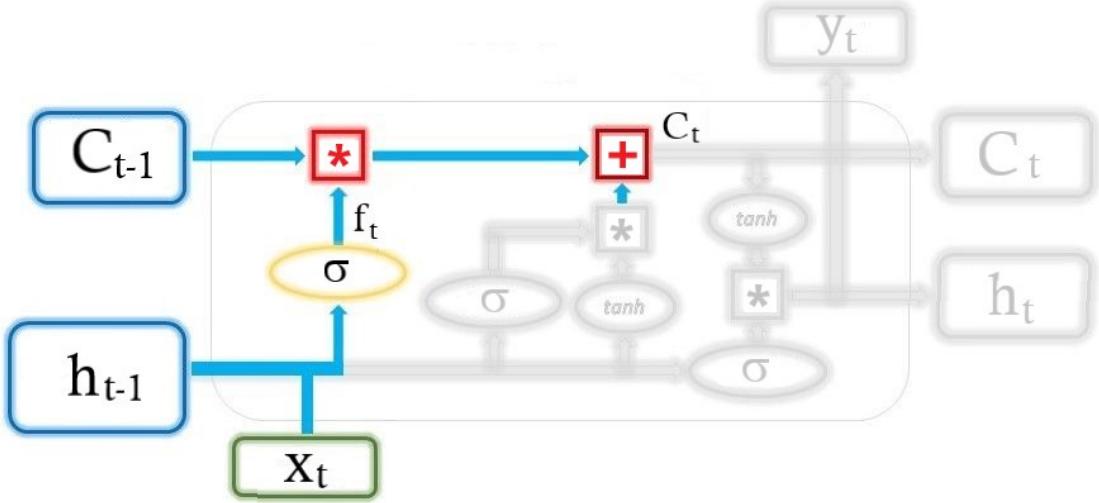


Рис. 13: Блок забывания ячейки LSTM

В блоке забывания происходит фильтрация долговременной памяти (какую часть следует сохранить, а какую отбросить). Это делается путем умножения поступающего состояния предыдущей ячейки на вектор забывания (f_t), который получается пропусканием текущего элемента входной последовательности и скрытого состояния предыдущей ячейки с соответствующими матрицами весов и смещением через функцию активации (сигмоиду).

$$f_t = \sigma(W_f \cdot (h_{t-1}, x_t) + W_{f_0})$$

Полученный вектор, состоящий из чисел в диапазоне от 0 до 1, будет умножен на состояние предыдущей ячейки (долгосрочную память), чтобы определить какие ее компоненты стоит сохранить.

Далее информация, полученная с входного блока и блока забывания суммируются и получается состояние текущей ячейки (текущее состояние долговременной памяти). Полученное состояние ячейки передается в следующую ячейку LSTM.

$$C_t = C_{t-1} * f_t + k_t$$

7.4 Выходной блок

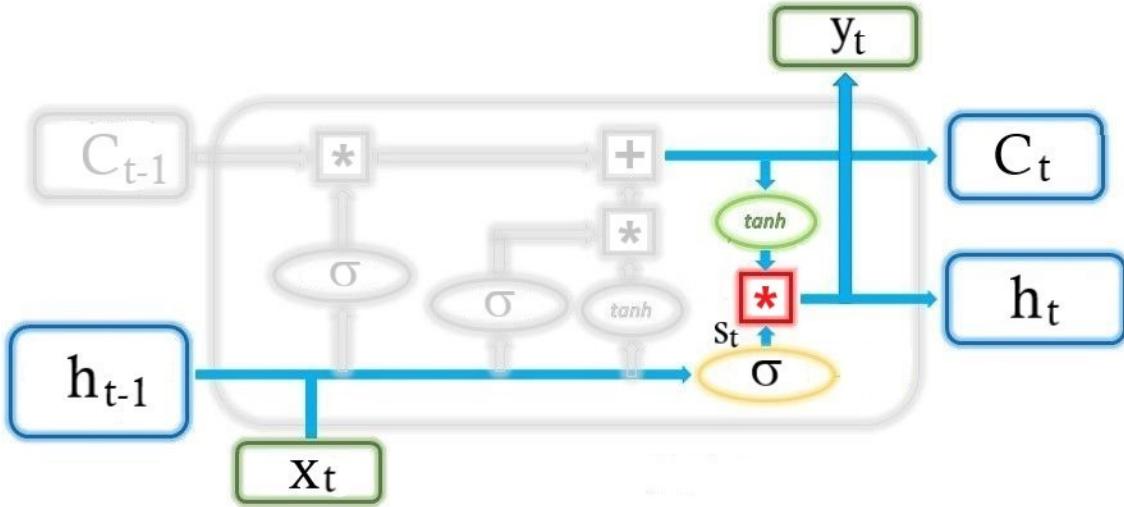


Рис. 14: Выходной блок ячейки LSTM

Выходной блок принимает текущий элемент входной последовательности, предыдущее скрытое состояние (предыдущую кратковременную память) и состояние текущей ячейки (текущую долговременную память) для создания текущего скрытого состояния, которое будет передано следующей ячейке (на следующем временном шаге). Элемент выходной последовательности, соответствующий текущему времениному шагу, так же может быть получен из этого скрытого состояния.

Скрытое состояние предыдущей ячейки и текущий элемент входной последовательности с соответствующими матрицами весов и смещением фильтруются сигмовидным слоем и суммируются с состоянием текущей ячейки, пропущенным через функцию активации $tanh$. В результате мы получаем скрытое состояние текущей ячейки.

$$s_t = \sigma(W_s \cdot (h_{t-1}, x_t) + W_{s_0})$$

$$h_t = s_t * \tanh(C_t)$$

Полученные состояние и скрытое состояние передаются на следующую ячейку LSTM. Соответствующий элемент выходной последовательности каждого временишага может быть получен из скрытого состояния текущей ячейки путем умножения на соответствующую матрицу весов и добавлением смещения:

$$y_t = W_y \cdot h_t + W_{y_0}$$

7.5 Борьба с затухающим градиентом

Напомним, что в рассмотренной ранее рекуррентной сети скрытое состояние и элемент в выходной последовательности в момент времени t вычисляются по формулам:

$$h_t = \tanh(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t)$$

$$y_t = W_{yh} \cdot h_t$$

При обучении сети используется алгоритм обратного распространения ошибки сквозь время. На шаге t необходимо посчитать 3 градиента: $\frac{\partial J_t}{\partial W_{hx}}$, $\frac{\partial J_t}{\partial W_{hh}}$, $\frac{\partial J_t}{\partial W_{yh}}$. Рассмотрим подробнее почему же возникает затухание/взрывание градиента на примере $\frac{\partial J_t}{\partial W_{hh}}$.

$$\frac{\partial J_t}{\partial W_{hh}} = \frac{\partial J_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \dots \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W_{hh}} = \frac{\partial J_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left(\prod_{t=2}^k \frac{\partial h_t}{\partial h_{t-1}} \right) \frac{\partial h_1}{\partial W_{hh}}$$

Все частные производные берутся тривиально, за исключением $\frac{\partial h_t}{\partial h_{t-1}}$, распишем ее:

$$\begin{aligned} \frac{\partial h_t}{\partial h_{t-1}} &= \tanh'(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t) \cdot \frac{\partial}{\partial h_{t-1}} [W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t] = \\ &= \tanh'(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t) \cdot W_{hh} \end{aligned}$$

Тогда исходный градиент примет вид:

$$\frac{\partial J_t}{\partial W_{hh}} = \frac{\partial J_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left(\prod_{t=2}^k \tanh'(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t) \cdot W_{hh} \right) \frac{\partial h_1}{\partial W_{hh}}$$

Произведение $\prod_{t=2}^k \tanh'(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t)$ при больших k стремится к 0, т.к. производная функции активации (гиперболический тангенс) положительна и не превосходит 1. Это и называется затуханием градиента.

Тогда:

$$\prod_{t=2}^k \tanh'(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t) \rightarrow 0$$

На некотором шаге $\frac{\partial J_t}{\partial W_{hh}} \rightarrow 0$ и матрица весов не будет обновляться. При последующих проходах будет наблюдаться аналогичная ситуация и обучение не будет эффективным.

Если элементы матрицы весов W_{hh} достаточно велики, чтобы подавить меньшую производную \tanh' , то возникает проблема взрывающегося градиента.

Теперь рассмотрим как была решена эта проблема в LSTM. Как было показано ранее основная проблема возникает с рекурсивным градиентом. Посчитаем $\frac{\partial C_t}{\partial C_{t-1}}$.

Напомним, что состояние ячейки LSTM в момент времени t определяется как:

$$C_t = C_{t-1} * f_t + j_t * i_t$$

А так же основные соотношения (смещения возьмем равные 0 для более краткой записи, т.е. $W_{f_0}, W_{j_0}, W_{i_0}, W_{s_0} = 0$):

$$\begin{aligned} f_t &= \sigma(W_f(h_{t-1}, x_t)) \\ i_t &= \sigma(W_i(h_{t-1}, x_t)) \\ s_t &= \sigma(W_s(h_{t-1}, x_t)) \\ j_t &= \tanh(W_j(h_{t-1}, x_t)) \\ h_t &= s_t * \tanh(C_t) \end{aligned}$$

Тогда

$$\begin{aligned}\frac{\partial C_t}{\partial C_{t-1}} &= \frac{\partial C_t}{\partial f_t} \frac{\partial f_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial i_t} \frac{\partial i_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} \\ &\quad + \frac{\partial C_t}{\partial j_t} \frac{\partial j_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial C_{t-1}} + \frac{\partial C_t}{\partial C_{t-1}}\end{aligned}$$

Возьмем частные производные правой части:

$$\begin{aligned}\frac{\partial C_t}{\partial C_{t-1}} &= C_{t-1} \sigma'(W_f(h_{t-1}, x_t)) W_f * s_{t-1} \tanh'(C_{t-1}) \\ &\quad + j_t \sigma'(W_i(h_{t-1}, x_t)) W_i * s_{t-1} \tanh'(C_{t-1}) \\ &\quad + i_t \tanh'(W_j(h_{t-1}, x_t)) W_C * s_{t-1} \tanh'(C_{t-1}) \\ &\quad + f_t\end{aligned}$$

При обратном распространении ошибки через k временных шагов у нас будет произведение градиентов посчитанных выше. При $k \rightarrow \infty$ произведение градиентов $\left(\prod_{t=2}^k \frac{\partial C_t}{\partial C_{t-1}}\right)$ не гарантирует схождение к нулю или бесконечности (как в случае обычной рекурентной сети). Если на каком то шаге мы замечаем, что произведение градиентов начинает стремиться к 0, то мы можем скорректировать матрицу весов W_f , которая изменит значение f_t и тем самым произведение приблизится к 1 (аналогичным образом можно поступить и с другими функциями и соответствующие им матрицами весов). Тем самым предотвращается либо значительно замедляется затухание градиента. Важно отметить, что значения f_t, s_t, i_t, j_t сеть учится устанавливать (в зависимости от текущего элемента входной последовательности и скрытого состояния). Таким образом сеть учится решать когда позволить градиенту затухнуть, а когда сохранить его, соответствующим образом устанавливая значения матриц весов.

8 Методика определения изменения направления ходьбы и пройденного расстояния

8.1 Исходные данные

Для обучения сети мы используем данные пешеходов, полученные со смартфонов (показания акселерометров и ДУС в осях смартфона), находящихся в карманах брюк и руке, а так же информацию, полученную с IMU (траектории движения испытуемого, которые мы слаживаем), установленных на стопах испытуемых. Пешеходы прошли треки различной длины (3 и 5 минут). Путем решения навигационной задачи для ИНС на данных, полученных с IMU, были получены истинные траектории, которые используются в ходе обучения сети. Также для увеличения количества треков были взяты данные из открытых источников (датасет OxIOD). Данные из датасета OxIOD устроены схожим образом, однако в качестве истинных траекторий используются реальные перемещения смартфона, фиксируемые десятью камерами расположенными по периметру помещения, в котором производились эксперименты.

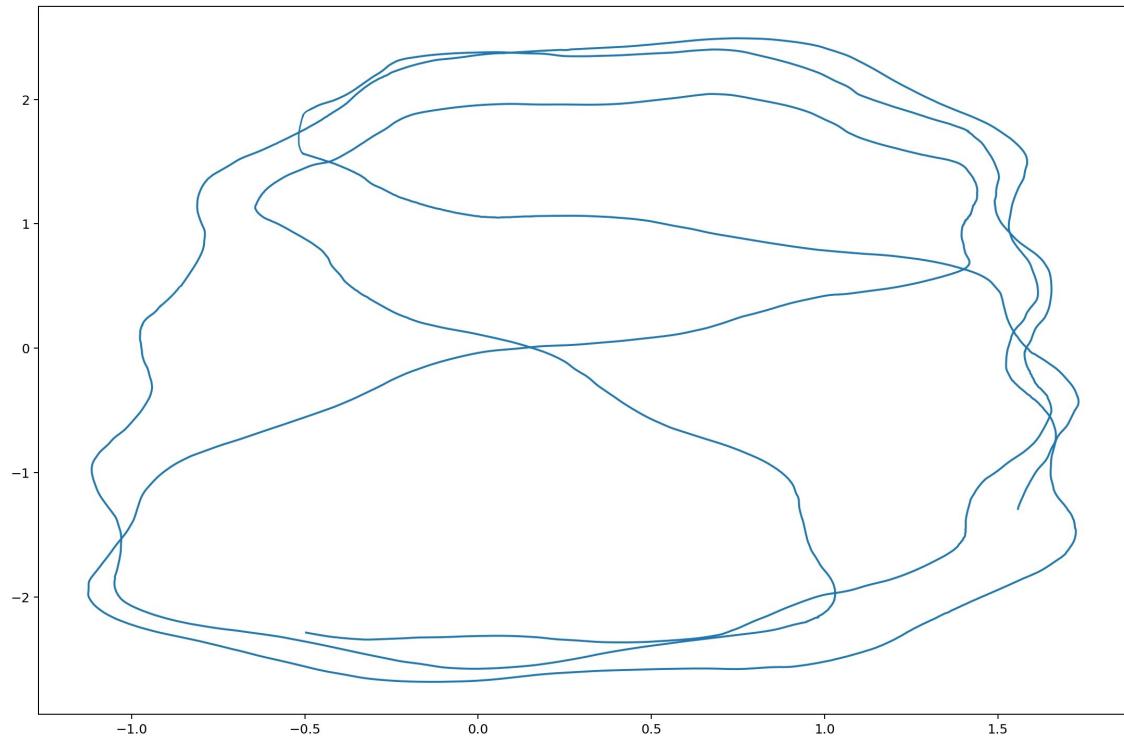


Рис. 15: Пример части траектории из открытых источников, используемой для обучения сети.

Поскольку, траектории из открытых источников получены другим методом они гладкие. Для единообразия данных для обучения мы сглаживаем имеющиеся траектории, полученные с IMU.

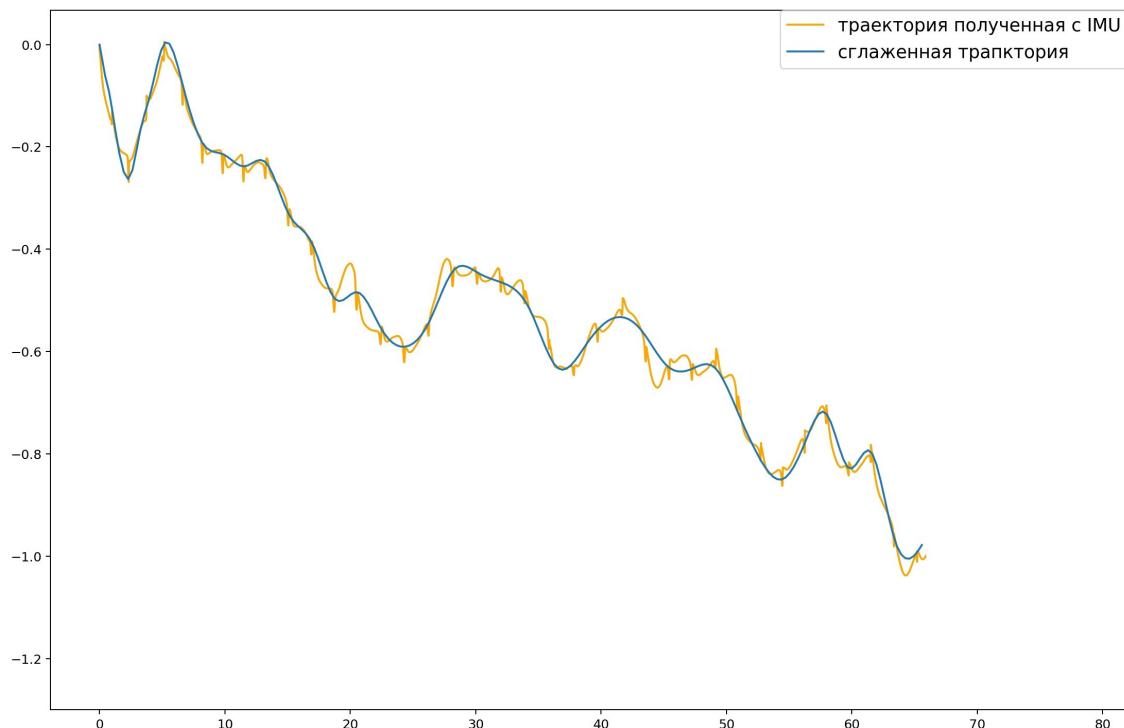


Рис. 16: Сравнение части траектории, полученной при помощи IMU и сглаженной траектории.

Для обучения будет использоваться 60 треков различной длины, а оставшиеся 5 треков используются для тестирования сети.

8.2 Описание используемых устройств

Имеющиеся данные были получены с помощью 3 телефонов. Телефон, находящийся в руке у испытуемого, Huawei VTR-L29 с блоком из акселерометра и гироскопа — lsm6dsm. Телефон, находящийся в правом кармане брюк у испытуемого, Xiaomi MI 6 с блоком из акселерометра и гироскопа — ICM20690. Телефон, находящийся в левом кармане брюк у испытуемого, Samsung SM-G950F с блоком из акселерометра и гироскопа — LSM6DSL.

В данных из открытых источников используется смартфон Iphone 7. Информация о модели блока акселерометров и гироскопа отсутствует.

Данные полученные со смартфонов (показания акселерометров и гироскопов) используются без дополнительной обработки.

8.3 Синхронизация данных

Рассмотрим пятиминутный трек ходьбы человека. У пешехода находятся 3 телефона, расположенные в левом и правом передних карманах брюк, а так же в руке. Со смартфона мы получаем сырье данные, которые необходимо преобразовать для дальнейшей работы. Данные для каждого из смартфонов находятся в двух файлах (показания акселерометров и ДУС), каждый из которых представляет собой матрицу m на n , где $n = 5$, а m равно количеству измерений. В столбцах расположены последовательно: время измерения (начало отсчета от старта запуска приложения), точность измерения и в последних трех столбцах - показания акселерометра либо показания датчиков угловой скорости.

Обычно запись показаний акселерометров и ДУС (в одном телефоне) начинается в разное время, поэтому за начало берем максимальное начальное время из двух файлов. Аналогично и с концами файла- берем минимальное время из двух файлов. Используем медианный фильтр для устранения нежелательных шумов.

Так как разные устройства имеют разную частоту записи, то их необходимо синхронизировать. Наша задача добиться частоты 100 Гц (запись каждые 0.01 секунд) на всех устройствах, в том числе и на IMU на стопах. Для этого создаем время с постоянным шагом и делаем интерполяцию ко времени.

8.4 Подготовка данных для обучения

Для обучения сети нам необходимы данные со смартфонов (показания акселерометров и ДУС), а так же траектории, полученные с IMU, расположенных на стопах. Входными данными для нейронной сети служат показания смартфонов на двухсекундном интервале времени (200 наборов показаний акселерометров и ДУС).

t_1	t_2	t_3		t_{95}		t_{105}		t_{200}
a_{1x}	a_{2x}	a_{3x}		a_{95x}		a_{105x}		a_{200x}
a_{1y}	a_{2y}	a_{3y}		a_{95y}		a_{105y}		a_{200y}
a_{1z}	a_{2z}	a_{3z}	• • •	a_{95z}	• • •	a_{105z}	• • •	a_{200z}
w_{1x}	w_{2x}	w_{3x}		w_{95x}		w_{105x}		w_{200x}
w_{1y}	w_{2y}	w_{3y}		w_{95y}		w_{105y}		w_{200y}
w_{1z}	w_{2z}	w_{3z}		w_{95z}		w_{105z}		w_{200z}

Рис. 17: Входные данные нейронной сети. На вход поступает 200 показаний акселерометров и ДУС (набор показаний $(a_{ix}, a_{iy}, a_{iz}, w_{ix}, w_{iy}, w_{iz})$ получен в таймстемп t_i).

Данные берутся с шагом в 10 временных отсчетов. То есть первый набор входных данных - это показания с 1 по 200 отсчеты, второй набор данных - это показания с 10 по 210 отсчеты и так далее. Выходными данными сети являются пройденное расстояния и изменение направления движения на участке за 10 отсчетов как продемонстрировано на рисунке 18.

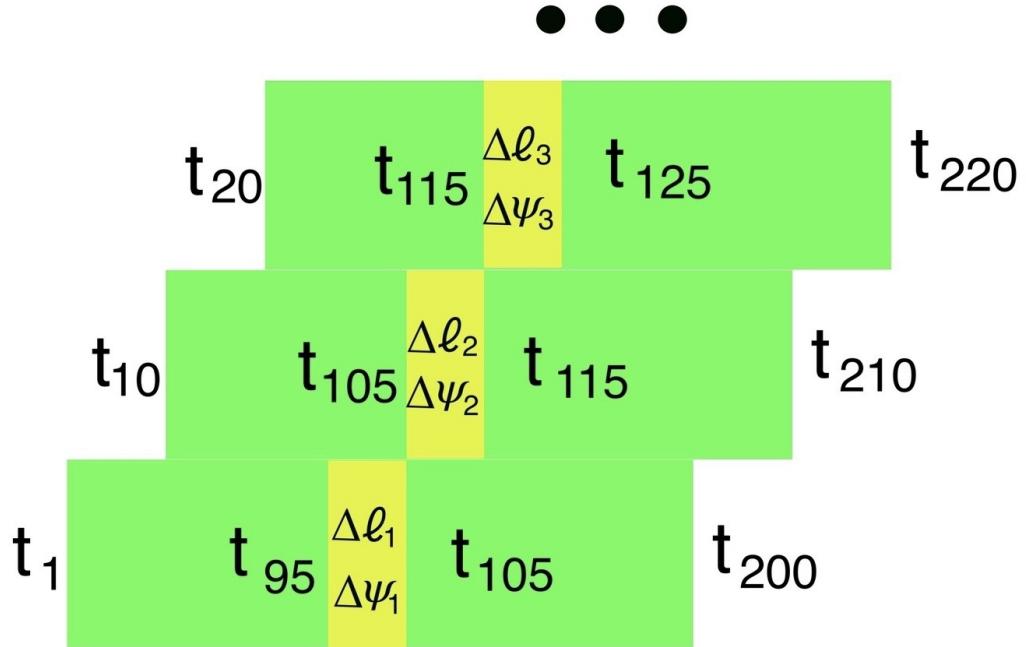


Рис. 18: Процесс формирования датасета из последовательно идущего набора показаний акселерометров и ДУС. Первый элемент датасета - это набор показаний акселерометров и ДУС взятые с t_1 по t_{200} , а так же пройденное расстояние (Δl) и изменение направления движения ($\Delta \psi$) с t_{95} по t_{105} .

8.5 Архитектура используемой в работе нейронной сети

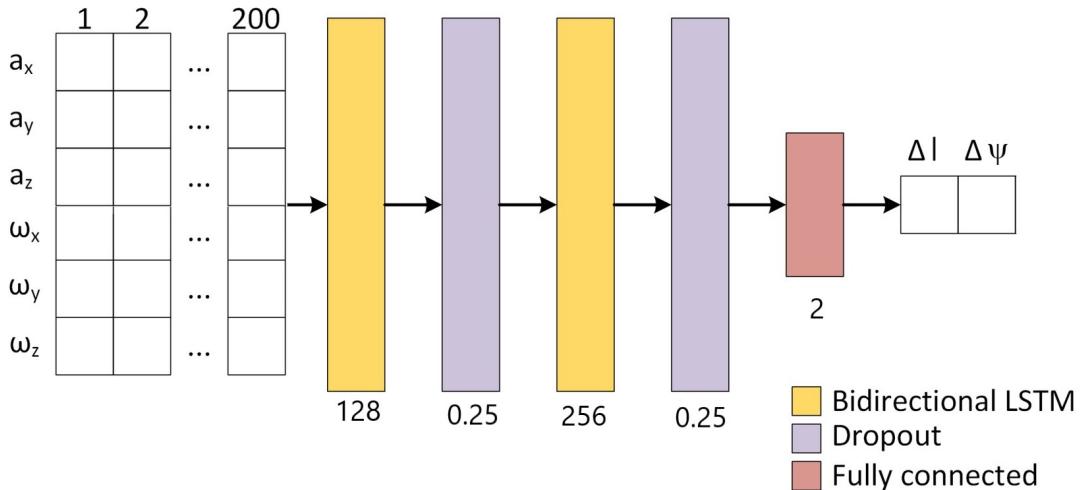


Рис. 19: Архитектура используемой в работе нейронной сети

На вход сети поступают 200 наборов акселерометров и ДУС, взятые последовательно на двухсекундном интервале времени. Каждый вектор из шести показаний проходит два полно связанных слоя двунаправленной *LSTM* со 128 и 256 скрытыми слоями на каждом. После каждого слоя *LSTM* применяется слой *Dropout* с параметром 25%, чтобы избежать переобучения (выбрасываем случайным образом 25% нейронов). Скорость обучения 0.0001. На выходе из сети мы получаем пройденное расстояние и изменение направления движения ($\Delta l, \Delta \psi$). Получается, что трек разбивается на двухсекундные независимые окна на выходе из каждого окна получаем $(\Delta l_i, \Delta \psi_i)$.

В качестве функции потерь выбирается:

$$J = \sum \|\Delta \tilde{l} - \Delta l\|_2^2 + l_*^2 \|\Delta \tilde{\psi} - \Delta \psi\|_2^2$$

, где $\Delta \tilde{l}$ и $\Delta \tilde{\psi}$ - истинные значения, параметр $l_* = 2$ м.

При обучении нейронной сети мы используем 60 треков различной длины и различной траектории. Данные разбиваются на обучающие подвыборки размером 64 (образуя последовательность из 64 наборов показаний смартфона и соответствующие им выходные данные, каждый набор - 200 последовательных показаний) и перемешиваются на каждой эпохе. Входными данными являются данные акселерометров и ДУС, а выходными - пройденные расстояния и изменения направления движения. При обучении мы используем $(\Delta l, \Delta \psi)$, полученные при помощи имеющихся точных траекторий.

При тестировании мы используем только данные смартфона и затем сверяем полученные результаты с результатами полученными по точным траекториям.

8.6 Результаты

Описанная ранее сеть была написана на основе библиотеки PyTorch и обучена на видеокарте GEFORCE RTX 2070 на 100 эпохах (одна эпоха - это проход всего

датасета через нейронную сеть). Одна эпоха в среднем занимает 5 минут при размере обучающей подвыборки равном 64. В качестве оптимизатора был выбран Adam (базируется на использовании алгоритма градиентного спуска).

Стоит отметить, что начальная выставка для смартфонов не производится и поэтому полученная траектория отличается от истинной на начальный угол курса.

Рассмотрим истинную и предсказанную траектории пользователя со смартфоном в руке (рис. 20):

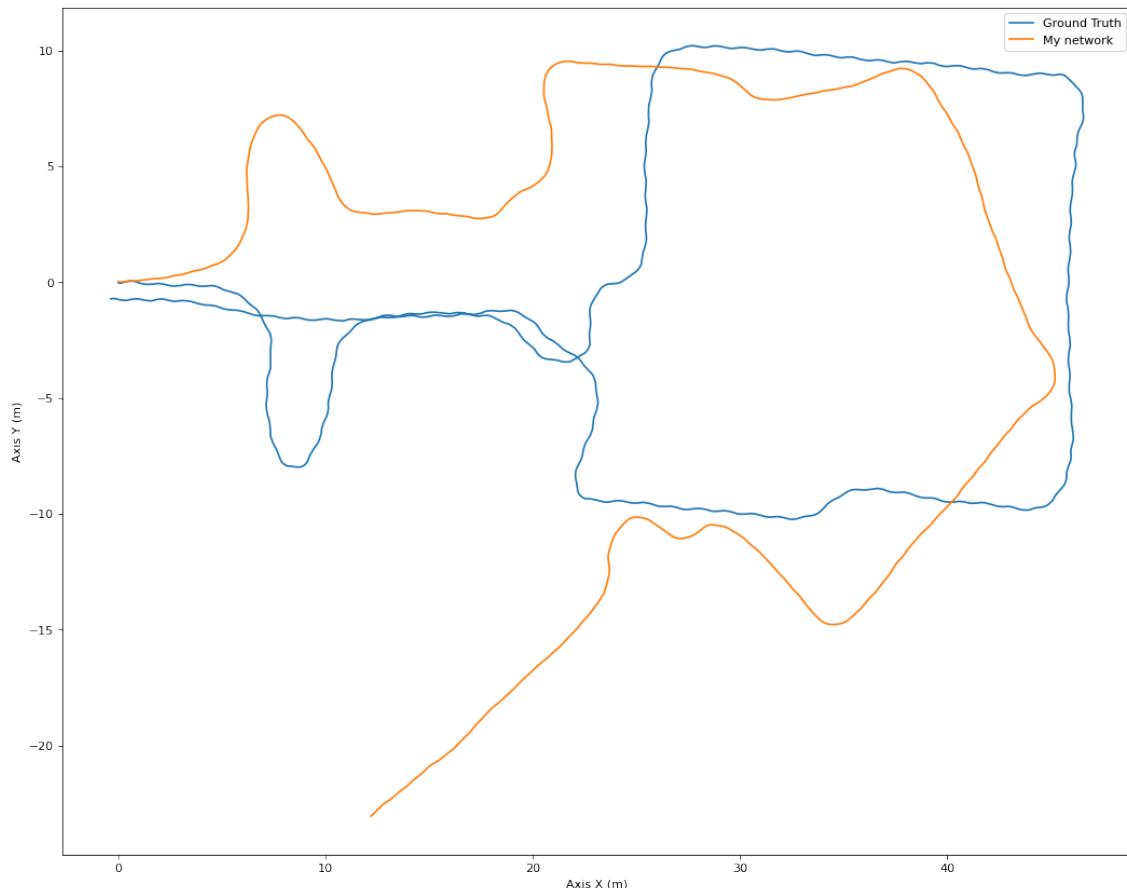


Рис. 20: Траектория, построенная при помощи нейронной сети (оранжевым цветом) и истинная траектория (синим цветом)

Теперь построим траекторию по данным смартфона в кармане брюк (рис. 21):

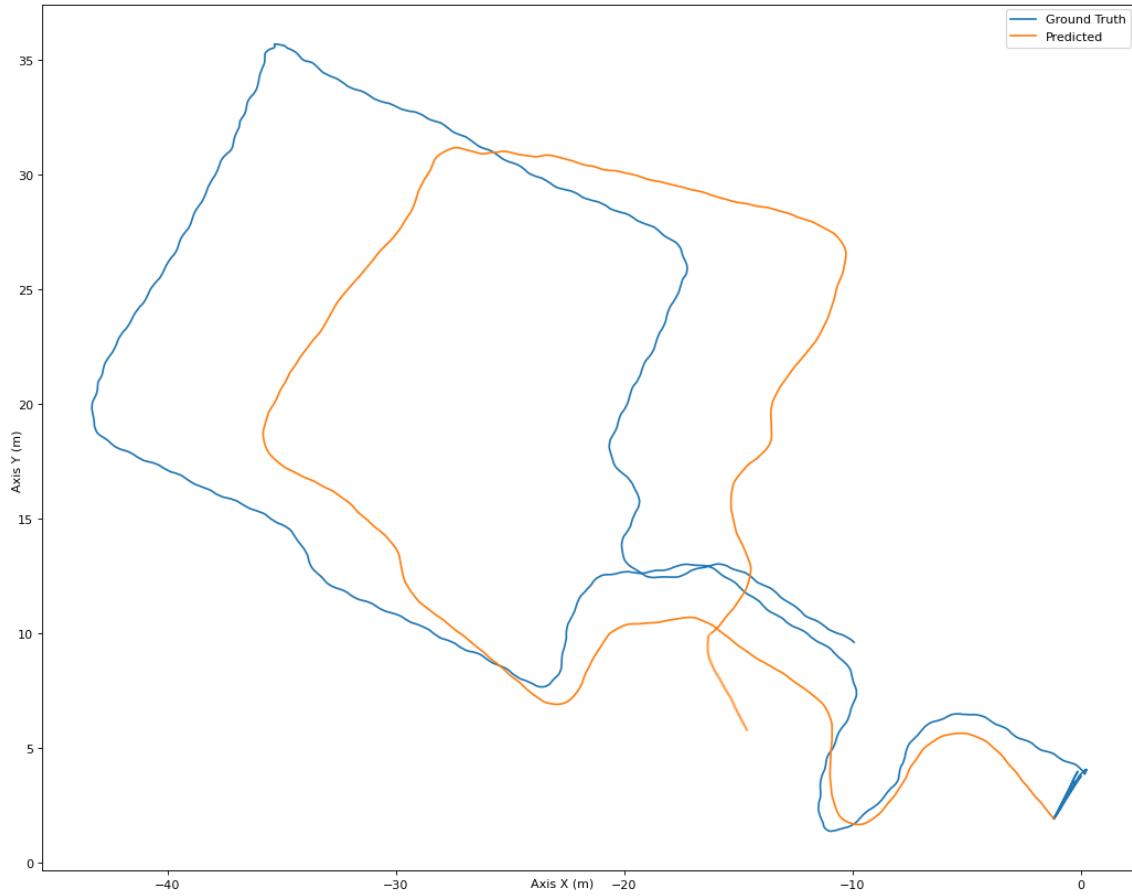


Рис. 21: Траектория, построенная при помощи нейронной сети (оранжевым цветом) и истинная траектория (синим цветом)

Теперь рассмотрим 5 минутный имеющийся трек ходьбы пешехода со смартфоном в кармане брюк. На рисунке 22 представлен результат работы сети, описанной ранее, в сравнении с истинной траекторией.

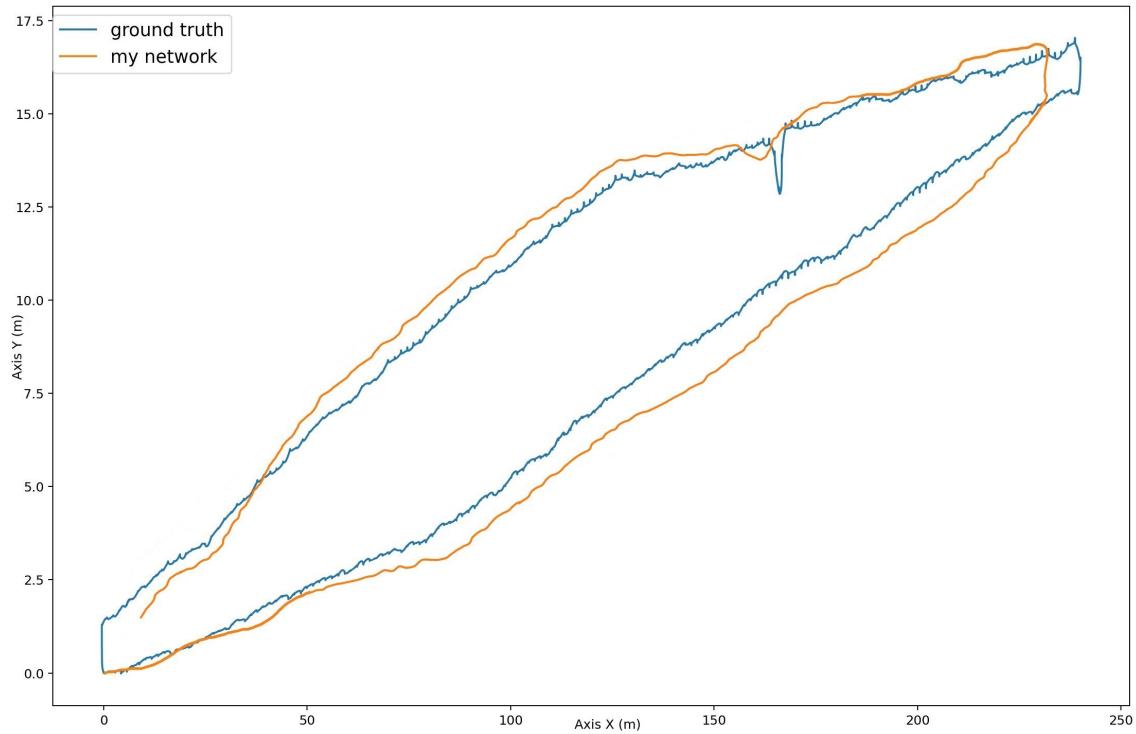


Рис. 22: Траектория, построенная при помощи нейронной сети (оранжевым цветом), и истинная траектория (синим цветом)

И еще один имеющийся трек с телефоном в руке (рис. 23):

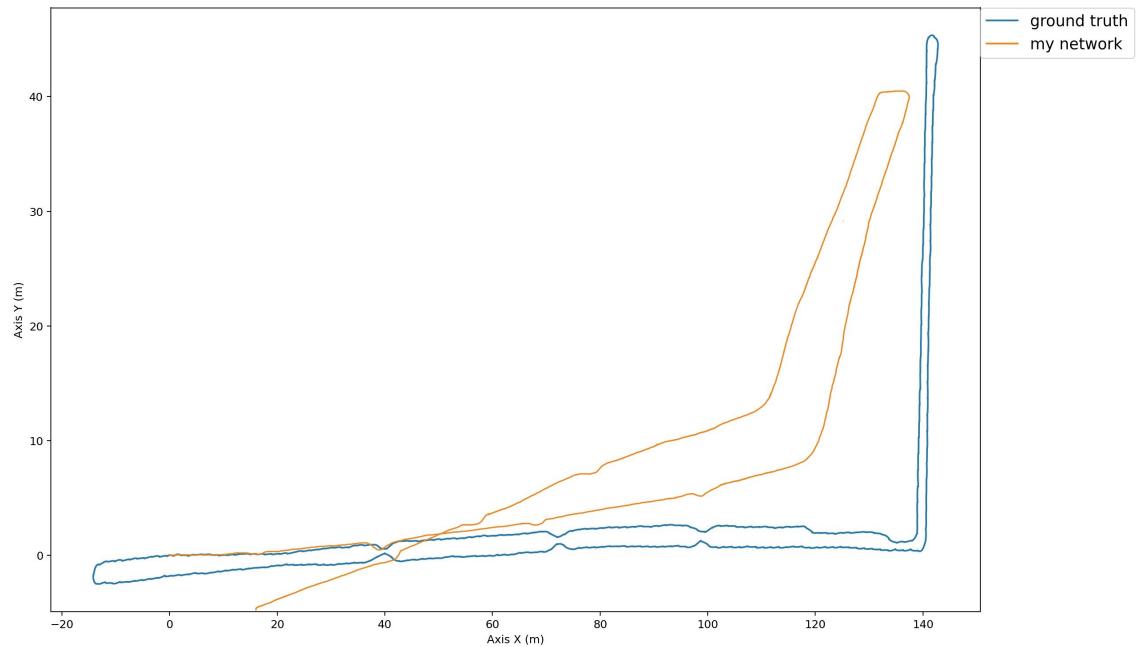


Рис. 23: Траектория, построенная при помощи нейронной сети (оранжевым цветом), и истинная траектория (синим цветом)

9 Навигация с использованием комбинации фильтра Мэджвика и метода главных компонент

9.1 Описание метода

Рассмотрим иной подход к решению данной задачи. Суть его заключалась в определении ориентации смартфона при поступлении информации с датчиков в каждый момент времени. Информацией являются показания акселерометра, ДУС и магнитометра. На каждом шаге мы определяем направление ходьбы и применяем эмпирическую формулы для подсчета длины шага.

Опишем алгоритм подробнее.

Сперва производится начальная выставка для определения начальной ориентации смартфона относительно опорной системы координат. В качестве опорной системы координат выбирается система, связанная с Землей. За исходную систему координат выбирается положение смартфона в начальный момент времени, когда телефон неподвижен. Когда смартфон неподвижен мы можем воспользоваться уравнением: $a^T = -L \cdot g$, где a - вектор измерений акселерометра, L - матрица перехода от опорного трехгранника к исходной системе координат, g - вектор ускорения свободного падения. Так мы определяем начальные значения углов крена и тангла, а угол курса полагаем равным 0.

Для определения текущей ориентации во время движения пешехода применяется фильтр Madgwick с использованием данных акселерометра, ДУС и магнитометра. Когда мы знаем проекции каждого ускорения на оси смартфона и матрицу поворота (получаем из ориентации), мы получаем проекции показаний акселерометров на исходную систему координат.

Для определения длины шага сперва определим что подразумевается под понятием "шаг". Будем считать, что шаг происходит с момента отрыва мыска ноги и до касания пяткой земли противоположной ноги (рис. 25).

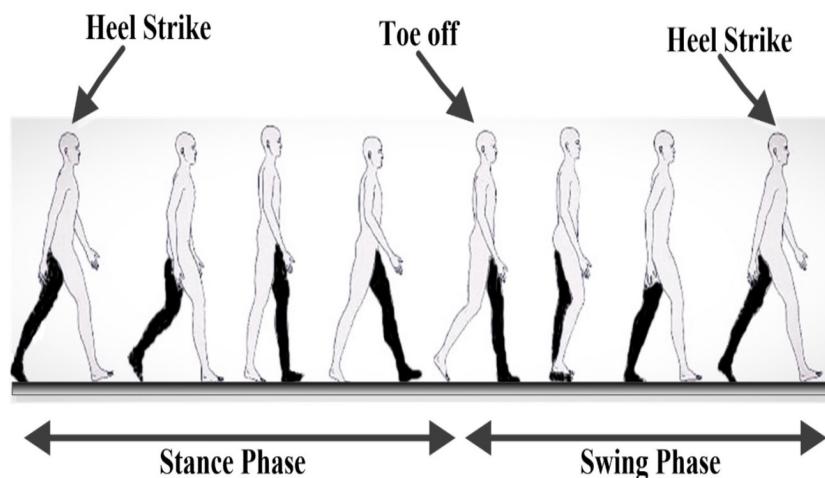


Рис. 24: Шаг включает две основные фазы: опоры и переноса

Во время шага наблюдается два пиковых момента. Один ярковыраженный - это

перенос ноги, в кармане которой находится смартфон и второй слегка слабее - перенос противоположной ноги (фаза опоры ноги со смартфоном).

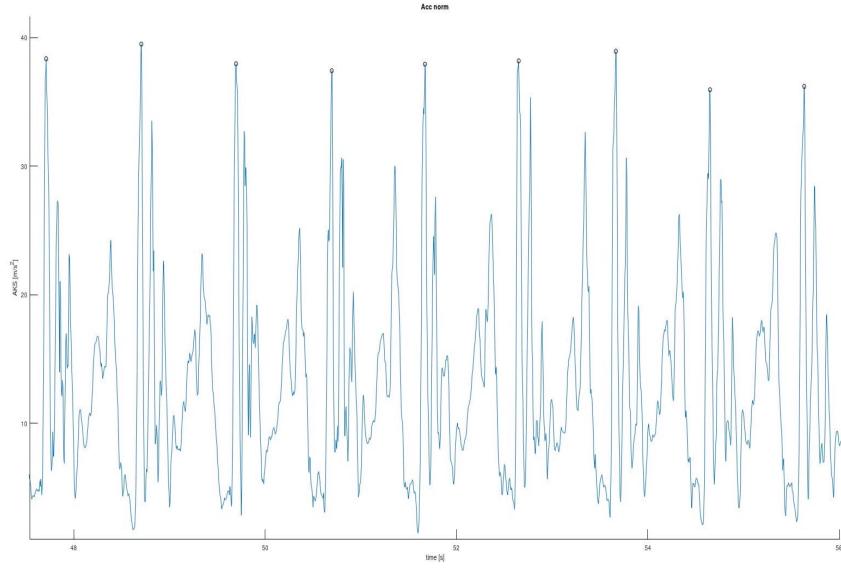


Рис. 25: График нормы показаний акселерометров. Ось ординат - норма показаний акселерометра (m/s^2), ось абсцисс - время (с)

Пользуясь эмпирической формулой [10] вида: $StepLen = K \sqrt[4]{A_{max} - A_{min}}$, где A_{max} и A_{min} - это максимальное и минимальное значения нормы ускорения за один шаг, а K - это параметр, характеризующий точность определения пройденного расстояния.

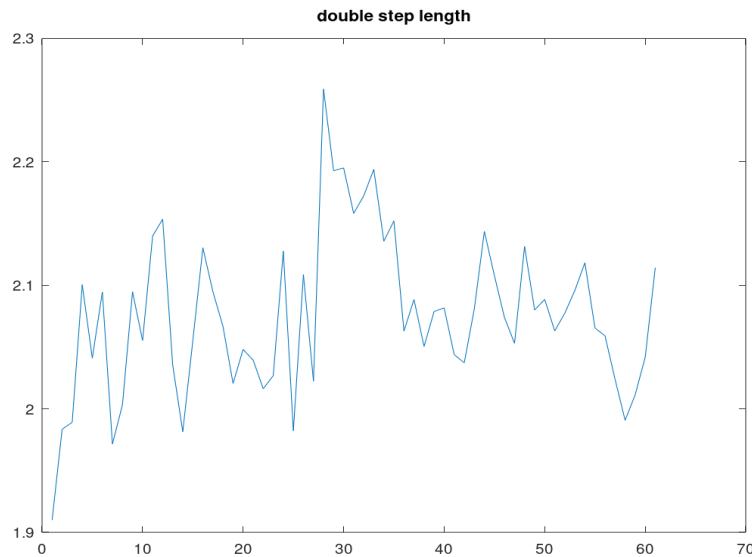


Рис. 26: Длины шагов за весь трек. Ось ординат - длина шага (м), ось абсцисс - время (с)

Во время ходьбы движение происходит в горизонтальной плоскости, поэтому мы будем применять РСА алгоритм [11] к проекциям наших данных на исходную си-

систему координат, а точнее на ее горизонтальную составляющую (плоскость). После применения РСА алгоритма мы получаем набор двумерных векторов главных компонент. Эти вектора и определяют направление ходьбы на каждом шаге. Зная длину шага и направление мы строим траекторию (умножая длину на направление).

9.2 Результаты

Рассмотрим результаты полученные при помощи РСА алгоритма и фильтра Madgwick:

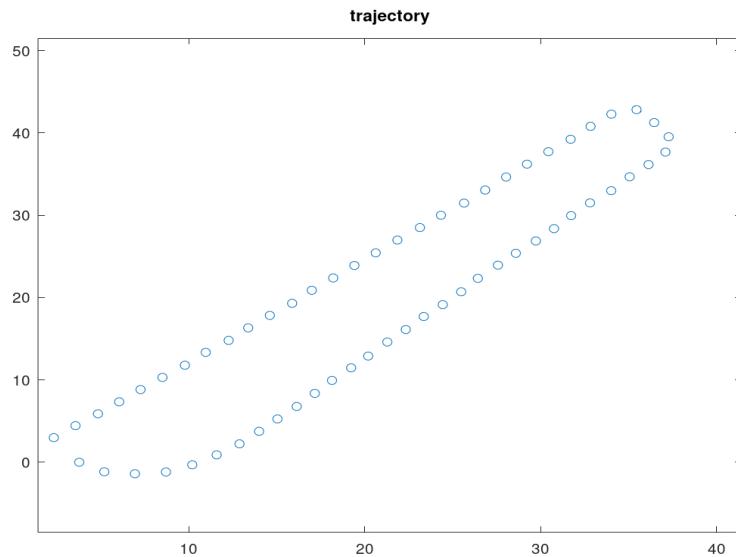


Рис. 27: Полученная траектория после применения РСА алгоритма к данным со смартфона, находящимся в правом кармане брюк, частота 200 Гц. Параметр β в фильтре Madgwick равен 0.05)

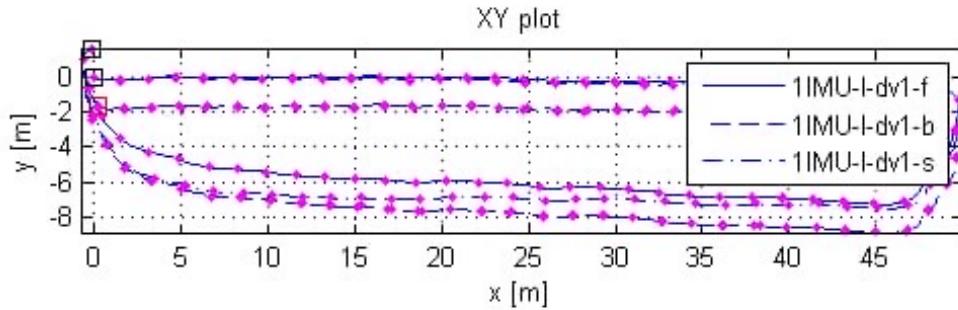
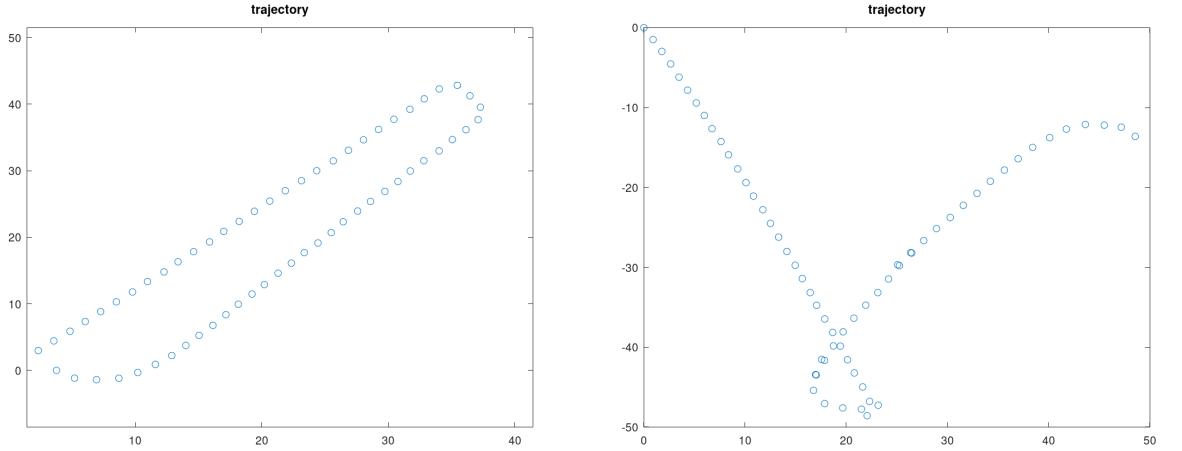


Рис. 28: Оценки траектории левой и правой ноги (данные с IMU)

9.3 Недостатки метода

К основным недостаткам данного метода стоит отнести необходимость в подборе параметра β в фильтре Мэджвика. Коэффициент усиления фильтра β представляет все ошибки измерений нуля гироскопа. Источники ошибки: шум датчика, сглаживающий фильтр, ошибки калибровки, ошибки установки и выравнивания датчика, неортогональность осей датчика и другие. Параметр β определялся индивидуально для каждого смартфона и если подобрать его неправильно, то получается неверная траектория. Еще один недостаток - это параметр K , который присутствует в эмпирической формуле для определения длины шага подбирается таким же образом (мы определили его из траектории, полученной с данных IMU).

Сравним траектории одного испытуемого с телефонами в правом кармане брюк, но с разными параметрами β :



Полученная траектория после применения PCA алгоритма к данным со смартфона, находящимся в правом кармане брюк, частота 200 Гц. Параметр β в фильтре Madgwick равен 0.05

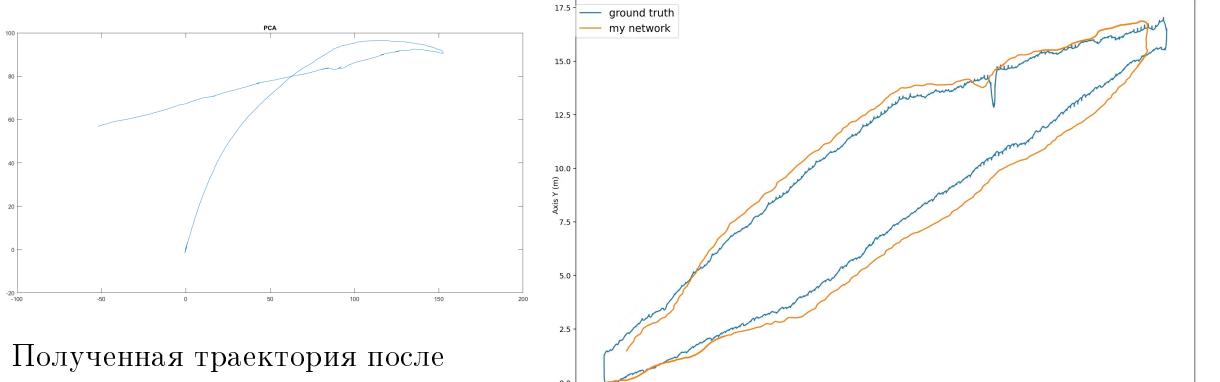
Полученная траектория после применения PCA алгоритма к данным со смартфона, находящимся в правом кармане брюк, частота 200 Гц. Параметр β в фильтре Madgwick равен 0.2

Рис. 29: Зависимость полученной траектории от параметра β .

Продемонстрированные траектории (рис. 30) соответствуют истинной траектории из прошлого раздела.

9.4 Сравнение методов

Построим траектории одного и того же смартфона, но полученные разными методами, а так же истинную траекторию (рис. 31)



Полученная траектория после применения PCA алгоритма к данным со смартфона, находящимся в правом кармане брюк. Параметр β в фильтре Madgwick равен 0.05

Построенная траектория при помощи нейронной сети (оранжевым цветом) и истинная траектория (синим цветом)

Рис. 30: Сравнение методов.

10 Выводы

Рассмотрены два метода автономной навигации с использованием инерциальных датчиков.

В ходе дипломной работы была построена сеть, которая строит траектории по показаниям акселерометра и ДУС, полученных со смартфона. Рассмотренный вариант решения данной задачи с использованием магнитометров страдает от проблем с определением точной ориентации, тк показания магнитометра имеют достаточно большой шум. Если иметь точную ориентацию смартфона в каждый момент времени, то можно построить более точную траекторию, но рассчитывать на точную ориентацию нельзя. Поэтому метод, основанный на использовании нейронных сетей, ограничивается лишь показаниями акселерометров и ДУС.

Однако, стоит отметить, что решающую роль в ходе реализации метода с использованием нейронных сетей играет обучающая выборка. Чем она больше и разнообразнее, тем лучше нейронная сеть обучится и будет точнее строить траектории.

Метод, основанный на комбинации фильтра Мэджвика и метода главных компонент показал менее стабильные результаты, хотя в нем помимо инерциальных датчиков использовались показания магнитометра.

Можно сделать вывод, что для автономной навигации с использованием смартфона методы машинного обучения на основе нейронных сетей являются предпочтительными.

Список используемой литературы

- [1] Steven Walczak, Narciso Cerpa - "Artificial Neural Networks Encyclopedia of Physical Science and Technology (Third Edition), 2003, pages 631-645.
- [2] Melody Y.Kiang - "Encyclopedia of Information Systems Boston : Academic Press, 2003, pages 303-315.
- [3] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, Andrew Markham, Niki Trigoni - "OxIOD: The Dataset for Deep Inertial Odometry Department of Computer Science, University of Oxford, September 2018.
- [4] Вавилова Н.Б., Голован А.А., Парусников Н.А. - "Математические основы инерциальных навигационных систем". Издательство МГУ. Москва - 2018.
- [5] Hochreiter, Sepp and Schmidhuber, Jürgen - "Long Short-term Memory". Journal "Neural computation". December 1997.
- [6] Du, K.-L and Swamy, M.N.s - "Recurrent Neural Networks Neural Networks and Statistical Learning, December 2014, (pp.337-353).
- [7] Bolotin Y., Bragin A., Gulevskiy D. "On Consistency of EKF for Pedestrian Dead Reckoning with Foot Mounted IMU," 2020 27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), 2020, pp. 1-10.
- [8] Barry J. Wythoff - "Backpropagation neural networks: A tutorial Chemometrics and Intelligent Laboratory Systems, volume 18, issue 2, February 1993, pages 115-155.
- [9] P. J. Werbos - "Backpropagation through time: what it does and how to do it," in Proceedings of the IEEE, vol. 78, no. 10, Oct. 1990, pp. 1550-1560.
- [10] Jahn, J.; Batzer, U.; Seitz, J.; Patino-Studencka, L.; Boronat, J.G. "Comparison and evaluation of acceleration based step length estimators for handheld devices". In Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Zurich, Switzerland, 15–17 September 2010; pp. 1–6.
- [11] I. T. Jolliffe - "Principal component analysis". Columbia University Libraries. New York 2004
- [12] Sebastian O.H. Madgwick - "An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays Technical Report x-io and University of Bristol: Bristol, UK, 30 April 2010
- [13] P. G. Savage - "Strapdown Inertial Navigation Integration Algorithm Design Part 1: Attitude Algorithms Journal of Guidance, Control, and Dynamics, Vol. 21, No. 1, 1998, pp. 19-28.
- [14] Bloesch M., Omari S., Hutter, M., Siegwart R. - "Robust visual inertial odometry using a direct EKF-based approach IEEE International Conference on Intelligent Robots and Systems, volume 2015-Decem, 2015, pp. 298–304.
- [15] Lymberopoulos D., Liu J., Yang X., Choudhury R. R., Handziski V., Sen, S. - "A Realistic Evaluation and Comparison of Indoor Location Technologies: Experiences and Lessons Learned IPSN 2015, pp. 178–189.

- [16] Leutenegger S., Lynen S., Bosse, Siegwart R., Furgale P. - "Keyframe-based visualinertial odometry using nonlinear optimization The International Journal of Robotics Research 34(3),2015, pp. 314–334.
- [17] Shu Y., Shin K. G., He T., Chen J. - "Last-Mile Navigation Using Smartphones Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom, 2015, pp. 512–524.
- [18] Xiao Z., Wen H., Markham A., Trigoni N. - "Lightweight map matching for indoor localization using conditional random fields 2014 International Conference on Information Processing in Sensor Networks, IPSN 2014, pp. 131–142.
- [19] Hilsenbeck S., Bobkov D., Schroth G., Huitl R., Steinbach E. - "Graph-based Data Fusion of Pedometer and WiFi Measurements for Mobile Indoor Positioning UbiComp 14, 2014, pp. 147–158.
- [20] Wang S., Clark R., Wen H., Trigoni N. - "DeepVO : Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 2043-2050.
- [21] Clark, R.; Wang, S.; Wen, H.; Markham A., Trigoni N. - "VINet : Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem AAAI Conference on Artificial Intelligence, 31(1), 2017.
- [22] Herath S., Yan H., Furukawa Y. - "RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, New Methods 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 3146-3152.