Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Інформаційно-аналітична система фітнес-клубу для визначення типу фізичної активності людини
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Воронін Б.С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник д.т.н., проф. Аврунін О.Г
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Вороніну Богдану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційно-аналітична система фітнес-клубу для визначення типу фізичної активності людини

затверджена наказом університету від 31 березня 20 23 р. № 306Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 20 23 р.

3. Вихідні дані до роботи Проектування та розробка компонентів інформаційно-аналітичної системи фітнес-клубу для визначення типу фізичної активності людини. Проведення аналізу предметної області, конкурентів з існуючими програмними рішеннями. Аналіз наукової літератури з визначенням кращих практик при реалізації схожих задач, та з метою визначення оптимальних технологій реалізації. Порівняльний аналіз можливих методів реалізації системи та визначення оптимального. Операційна система – MacOS Ventura 13.3.1, програмне забезпечення: IDE-засіб Pycharm, IDE-засіб xCode, Jupyter Notebook

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної області та бізнес-процесів які можна автоматизувати. Розробка вимог до інформаційно-аналітичної системи фітнес-клубу для визначення типу фізичної активності людини. Визначення вимог до вхідних даних класифікатора. Аналіз можливих методів реалізації системи. Порівняльний аналіз визначених альтернатив реалізації. Розробка системи класифікації типу фізичної активності людини та аналіз метрик ефективності та швидкодії.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1 – графічне представлення GRU блоку, Рисунок 2 – графічне представлення LSTM блоку; Рисунок 3 – архітектура одношарової мережі, Рисунок 4 – архітектура двошарової мережі, Рисунок 5 – графік метрик при порівнянні правил сегментації, Рисунок 6 – графік результатів тестування різної кількості нейронів на кожному шарі, Рисунок 7 – графік залежності точності системи від частоти дискретизації вхідних даних за різних архітектур, Рисунок 8 – графік залежності точності системи від значень гіперпаметрів системи

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Визначення проблеми	03-05.03.2023	Виконано
2	Проведення аналізу предметної області	06-16.03.2023	Виконано
3	Аналіз реалізованих систем конкурентів	16-19.03.2023	Виконано
4	Визначення основних бізнес-процесів фітнес-клубу	20-24.03.2023	Виконано
5	Дослідження наукової літератури	25.04-08.04.2023	Виконано
6	Розробка вимог до проектованої системи	08-12.04.2023	Виконано
7	Визначення вимог до вхідної інформації	12-15.04.2023	Виконано
8	Розробка інструменту по збору даних	15-17.04.2023	Виконано
9	Збір показників сенсорів та форматування даних	17-25.04.2023	Виконано
10	Проектування нейромережевого класифікатора	16-20.04.2023	Виконано
11	Аналіз метрик оцінки ефективності роботи класифікатора	20-23.04.2023	Виконано
12	Аналіз можливих інструментів розробки	23-24.04.2023	Виконано
13	Програмна реалізація класифікаторів за різними методами	24.04-01.05.2023	Виконано
14	Аналіз отриманих метрик	01-05.05.2023	Виконано
15	Оформлення пояснювальної записки	05-12.05.2023	Виконано

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ д.т.н., проф. Аврунін О.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 96 с., 11 рис., 2 табл., 3 дод., 16 джерел.

ГЛИБИННЕ НАВЧАННЯ, ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА, КЛАСИФІКАТОР, НЕЙРОННА МЕРЖА, ФІЗИЧНА АКТИВНІСТЬ, ФІТНЕС-КЛУБ, MACHINE LEARNING, RNN.

Об’єкт дослідження – фізична активність людини під час занять у фітнес-клубі, патерни рухів при виконанні окремих вправ.

Мета роботи – визначити оптимальний підхід до вирішення задачі класифікації фізичної активності людини з використанням сенсорів особистих пристроїв людини – а саме акселерометр та гіроскоп смарт-годинника та смартфону.

Методи дослідження – фіксуванням показників просторових сенсорів для проведення моделювання та проектування класифікатора фізичної активності людини, з визначенням кращих альтернатив проведенням порівняльного аналізу розроблених варіантів.

Було проведено аналіз наукової літератури з метою виявлення найкращих практик та основних патернів при проектуванні та розробці системи класифікації рухів людини. Було проведено аналіз конкурентних систем, визначені їх недоліки та визначені шляхи по їх уникненню. У результаті досліджень було розроблено класифікатор типу фізичної активності людини з точністю майже 93% при аналізу сегменту даних тривалістю 4 секунди при частоті дискретизації сигналу 25 Гц.

Проектована система надасть можливість закладам надання спортивних послуг покращити якість надання свої послуг, підвищити ознайомленість людей відносно фізичної культури та потенційно збільшить інтерес до активного способу життя, що добре вплине на загальне здоров’я людей.

ABSTRACT

Explanatory note: 96 p., 11 fig., 2 tabl., 3 ann., 16 sources.

CLASSIFIER, DEEP LEARNING, FITNESS CLUB, INFORMATION-ANALYTICAL SYSTEM, MACHINE LEARNING, NEURAL NETWORK, PHYSICAL ACTIVITY, RNN.

Research object – human physical activity during fitness club training, movement patterns during the performance of individual exercises.

The aim of the work is to determine the optimal approach to solving the problem of human physical activity classification using personal device sensors – namely, the accelerometer and gyroscope of a smartwatch and smartphone.

Research methods – recording spatial sensor indicators for modeling and designing a human physical activity classifier, determining the best alternatives by conducting comparative analysis of developed options.

A scientific literature analysis was conducted to identify best practices and patterns in designing and developing a human movement classification system. An analysis of competitive systems was conducted, their shortcomings were identified, and ways to avoid them were determined. As a result of the research, a human physical activity type classifier was developed with an accuracy of almost 93% when analyzing a data segment of 4 seconds at a signal sampling frequency of 25 Hz.

The designed system will allow sports facilities to improve the quality of their services, increase people's awareness of physical culture, and potentially increase interest in an active lifestyle, which will have a positive impact on people's overall health..

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Гц – герци;

IAC – інформаційно-аналітична система;

ФА – фізична активність;

DT – decision tree – дерева рішень;

GPS – global positioning system – система глобального позиціонування;

GRNN – Gated recurrent neural network – рекурентні нейронні мережі з керованими затворами;

GRU – Gated recurrent unit – рекурентний блок з керованими затворами, або керований рекурентний блок;

ISP – In-System Programming – внутрішньо системне програмування;

LSTM – Long short-term memory – довга коротко-строкова пам’ять;

MF1 – macro F1 score – макро F1 оцінка;

ReLu – Rectified Linear Unit – блок лінійної функції випрямлення;

RNN – Recurrent neural network – рекурентна нейронна мережа.

ВСТУП

На сьогоднішній день налічується чимало спорт-центрів та фітнес-клубів які надають людям різні послуги зі сфери активного образу життя. Дана галузь стає все більш актуальною у молодого покоління, яке насамперед є рушієм змін та прогресу серед інформаційних технологій.

Серед відвідувачів різних спортивних закладів та ентузіастів, які займаються вдома, все більш популярними становляться програмні засоби моніторингу та логування власних тренувань з метою формування статистики, яка буде одночасно нагадувати користувачам про вже пройдені етапи тренування, що буде мотивувати на продовження тримання певного темпу тренування для досягнення особистих цілей. Але головною проблемою більшості існуючих засобів є необхідність власноруч обирати тип активності яким людина планує зараз займатись, та необхідність власноруч вимикати процес відстеження активності. У такому випадку якщо людина почала виконання якоїсь вправи і власноруч не увімкнула відстеження активності в окремому програмному додатку, то сам процес та мета цього відстеження зникають, адже отримана статистика буде не повною, через що втрачає семантичне навантаження

Таким чином, проблемою кожного додатку є потенційна можливість пропустити певні досягнення користувача, що може бути вирішено окремою підсистемою автоматичного визначення типу фізичної активності людини, що, у свою чергу, надасть інструмент автоматичного протоколювання тренувань користувача без необхідності ручного управління початку-кінця заняття.

Таким чином, виконання даного проекту по розробці системи автоматичного визначення фізичної активності людини дозволить у певному обсязі звільнити користувача від зайвої роботи по ручному старті відстеження виконання вправ, що дасть змогу користувачеві сконцентруватися на виконанні вправи, а не на її відстеженні.

1 ОГЛЯД І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області, що визначається діяльністю підприємства

Фітнес-клуб – це певне місце яке включає тренажерний зал, кардіо-зону, зали аеробіки тощо. Як правило до фітнес-клубу необхідно придбати абонемент який визначає які саме послуги буде надано тому чи іншому відвідувачу. Здебільшого у мережевих клубах є власні програмні додатки для інформування клієнтів щодо різних подій у закладі, перегляду розкладу групових занять, надання індивідуальних знижок тощо. Таким чином під час відвідування закладу клієнт максимум використовує програмний застосунок клубу лише при реєстрації при вході чи виходу з клубу. Для протоколювання виконаних тренувань та моніторингу показників організму, клієнти, як правило, використовують інші програмні засоби та клієнтські застосунки, які ніяким чином не пов’язані з додатком закладу який вони відвідують.

Додавши до існуючого програмного додатку фітнес-клубу або створивши окрему систему, що містить елемент моніторингу фізичних показників людини та можливість протоколювання тренувань, заклад та бізнес загалом матимуть змогу здійснювати більш детальний моніторинг та аналіз поведінки клієнтів у своїх закладах. Це дозволить більш гнучко проводити маркетингові компанії, заохочувати клієнтів до замовлення додаткових послуг бізнесу тощо.

Однак у такому випадку з’являється питанням конкуренції програмних засобів по моніторингу фізичних показників користувачів. Щоб залучити клієнта до використання програмного засобу окремого клубу, цей засіб повинен мати щось, що відрізняє його від конкурентів. Такою функцією може стати автоматичне визначення ФА клієнта, за допомогою якого можна надавати рекомендації щодо покращення стану здоров'я та

фізичної форми та вести автоматичне детальне протоколювання. Крім того, можна розглянути можливість підключення до системи додаткових інформаційних джерел, таких як рекомендації щодо харчування та інші, щоб забезпечити комплексний інформаційний сервіс для клієнтів. Таким чином, програмний засіб фітнес-клубу стане не просто інструментом моніторингу, а повноцінним сервісом, який надасть клієнтам всю необхідну інформацію для досягнення їхніх фітнес-цілей та підтримки здорового способу життя.

1.2 Функціональний аналіз системи класифікації фізичної активності людини

Розглядаючи ІАС для забезпечення автоматизації необхідних бізнес-функцій фітнес-клубу можна визначити наступну її архітектуру.

Першим рівнем програмного забезпечення буде слугувати web застосунок у вигляді клієнт-серверного додатку який буде містити у собі промо-інформацію закладу, та функціонал за допомогою якого будуть автоматизовані певні бізнес-функції в залежності від особливостей бізнес-процесів окремого клубу, серед яких може бути, наприклад, придбання абонементу онлайн, знайомство та обрання особистого тренера, перегляд та запис на групові заняття тощо.

Другим рівнем ІАС можна вважати клієнтський додаток у смартфоні, який може слугувати персоналізованою карткою клієнта закладу для перепустки, з валідацією стану абонементу, можливістю протоколювати власні тренування та планувати розклад власних тренувань. Через цей рівень заклад може здійснювати особисту рекламу окремих послуг клубу та надання індивідуальних пропозицій в залежності від поведінки клієнта.

Третім рівнем можна вважати додаток у смарт-годиннику, який буде представляти собою додатковий клієнтський додаток до додатку ІАС у смартфоні, який може виконувати окремі функції, серед яких наприклад надання можливості реєстрації початку/кінця тренування, збір інформації

сенсорів пристрою, пуш повідомлення щодо самого процесу тренування тощо. Саме до цього рівня програмного забезпечення закладу є сенс імплементувати підсистему для автоматичного визначення типу ФА клієнта, і, можливо, сам процес по класифікації типу ФА та часткового протоколювання результатів класифікації.

Саму систему визначення ФА людини можна використовувати різними способами, наприклад такими як: надання інформації для окремих підсистем асистентів тренування чи тренерів, для кращого керування процесом тренування або надання необхідної інформації для програмного застосунку фітнес-клубу для загальної оцінки продуктивності тренування клієнта з метою формування оптимального графіку відвідування закладу для зменшення ризику різних травм. Тобто дана підсистема слугує універсальним інтерфейсом доступу до протоколу фізичної активності людини, який може бути використаним іншими підсистемами у своїх цілях.

Є незліченна кількість прикладів реалізованих систем, які використовують функцію протоколювання ФА людини, щоб допомогти користувачам відстежувати їхні тренування. Samsung Health, наприклад, пропонує інструмент по відстеженню активності людини впродовж дня і окремого тренування, але є й інші програми, які повністю зосереджені на окремих видах спорту: наприклад для бігу та ходьби існують спеціальні додатки Nike+ і Endomondo, для їзди на велосипеді та плавання існує додаток Strava. Незважаючи на те, що ці програми розпізнають дуже обмежену кількість дій, і зазвичай навіть при наявності підсистеми автоматичного виявлення виду ФА, ця підсистема має дуже обмежений функціонал та кількість ресурсів, і слугує виключно для нагадування користувачеві власноруч увімкнути відповідне тренування у додатку. Але ці додатки дають чудові результати, будучи надзвичайно точними у визначенні типу ФА на які вони акцентовані, пропонуючи своїм клієнтам такі переваги, як автоматична пауза, коли вони виявляють, що користувач

більше не виконує відстежувану вправу, і персоналізована аналітика розвитку фізичних показників.

Крім того, в останні роки спостерігалося значне зростання кількості смарт-годинників і фітнес-браслетів, таких як Fitbit і Apple Watch, які можуть відстежувати кількість кроків, режим сну, пасивні періоди тощо. Такі пристрої почали використовувати як компоненти складніших систем, які використовують багато датчиків для виконання поглибленого розпізнавання активності для таких сценаріїв, як охорона здоров’я, здорове старіння, переконлива технологія здорової поведінки тощо.

1.3 Аналіз підходів до реалізації класифікатора

Виходячи з існуючих статей та досліджень на тему вирішення задач визначення типу ФА людини, було використано чимало різних підходів з використанням різних математичних моделей. Основними можна назвати наступні: дерева рішень [1], нейронні мережі [2], класифікатор на основі алгоритму k-найближчих сусідів [3], класифікатор нечіткої логіки [4] та класифікатори на основі правил [5]. Кожен з наведених методів реалізації класифікаторів відрізняється один від одного своєю складністю, точність, універсальністю та гнучкістю.

Класифікатори на основі дерева рішень (decision-tree або DT) самі популярні серед досліджень де було використано лише один класифікатор. Після вилучення та вибору відповідних ознак з навчальних даних DT-алгоритм виводить пороги, що призводять до дихотомічного розбиття дерева, на основі похідних ознак. Недоліком цього методу є те, що якщо ознаки які були вибрано некоректно і мають високу інваріантність серед представників різних класів, ризик перекриття значень різних типів ФА збільшується, що призводить до зниження продуктивності класифікації через відносно простий алгоритм виведення порогів. З іншого боку, перевагами цього методу є низькі обчислювальні вимоги та проста

реалізація. Оскільки DT можна перетворити на графічне представлення, класифікатори на основі цього алгоритму підлягають розумінню та інтерпретації, ніж інші методи класифікації ML.

Класифікатори нейронних мереж або штучні нейронні мережі займають друге місце за популярністю серед знайдених досліджень, що використовували лише один класифікатор. Однак, на відміну від дерев рішень, користувачам складніше розуміти, як саме система класифікує вхідні дані. Класифікатори нейронних мереж – потужний підхід, який має потенціал для виявлення різних поз і рухів з високими показниками класифікації [2] . Одними з обмежень цього класифікатора є висока витрати на проектування на розробку системи, та високі обчислювальні витрати під час роботи системи. Також слід враховувати високу різноманітність підходів до реалізації нейронних мереж. Різні архітектури мережі можуть давати різний результат ефективності в залежності від різного числа та типів діяльності. Зі збільшенням кількості діяльності, тобто класів, результативність мережі з використанням одновимірного сенсору (а саме акселератор) знижувалася на 14% [6], тоді як у роботах з використанням більшої кількості сенсорів та їх більшої просторовості точність класифікації навіть з немалим набором класів може становити більше 95% [7]

Класифікатор k-найближчих сусідів (k-NN) є одним з найпростіших алгоритмів машинного навчання. Він призначає окремий екземпляр до класу, до якого належать більшість схожих за вхідними даними екземплярів. Однак обмеженням цього методу є знаходження найкращої кількості сусідів (k), яка є константою, визначеною користувачем. Крім того, порівняно з DT, k-NN може бути повільнішим, коли обробляється великий набір даних через складність обчислень відстаней екземплярів від центру класів.

Класифікатор на основі нечіткої логіки принципово кращий метод для реального життєвого визначення ФА людей, ніж традиційні підходи, оскільки вони можуть працювати з неоднозначністю вхідних даних [4]. Однак визначення відповідних нечітких функцій приналежності та нечітких

правил є складним завданням. При використанні нейро-нечіткого класифікатора [8], який, на відміну від звичайних кластеризаторів, не вважає межі між сусідніми класами жорсткими, а припускає, що переходи є неперервними, де об'єкт всередині області перетину має ступінь належності до кожного сусіднього класі. Було досягнуто точності класифікації на рівні 70% для визначення різних швидкостей ходьби за допомогою даних акселерометра; з додаванням даних серцевого ритму точність класифікації збільшилася до 99,03%. Однак жодне з відповідних досліджень не використовувало цей класифікатор для визначення позицій.

Основною особливістю класифікатора на основі правил для визначення типу ФА є високі витрати під час виведення цих самих правил за якими буде відбуватися класифікація. Тобто вхідних сигнал у вигляді даних з сенсорів на тілі людини буде потрапляти до системи і у результаті проходження сигналом певних порогів буде відбуватися класифікація. Самі пороги чи правила можуть бути визначені з загальної області знань чи з інших експертних систем. Великим недоліком даного підходу до реалізації класифікатора можна визначити складність додавання нових класів до системи, адже з підвищенням кількості класів, кількість порогів буде багатократно зростати, що на певному рівні призведе до зверхньої складності. Отримані дослідженнями результати, для яких використовувалися реальні дані, виявились досить успішними в класифікації типу ФА, наприклад: зміна стани людини, ходьба з різною швидкістю, лежання, сидіння, стояння і підйому по сходах. Але для більше складних систем які можуть мати динамічну структуру класів такий метод не буде ефективним.

1.4 Характеристики та вимоги до вхідних даних системи

Одним з основних викликів проектування та аналізу класифікаторів ФА людини є необхідність великої кількості позначених даних. Самі дані

представляють з себе інформацію зібрану з певних сенсорів вмонтованих у особисті пристрої людини, або зі спеціалізованих пристроїв. Зібрані дані мають високу ступінь зв’язаності, тобто здебільшого вони мають семантичну цінність лише якщо розглядати масив даних за певний проміжок часу, адже лише таким чином можна визначити динаміку рухів окремих частин тіла людини, де знаходяться слідкуючі пристрої.

Серед кращих практик, отриманих в результаті аналізу наукової літератури, для роботи з масивами даних з сенсорів зібраних з метою визначення ФА є новий метод [9], який вимагає лише невеликої кількості позначених даних для задач визначення типу ФА. Персоналізований метод заснований на пошуку похідних елементів даних між групою попередніх користувачів та цільовим користувачем. Порівнюючи персоналізований метод з загальною моделлю та користувацькою моделлю, персоналізований метод був кращим, коли була доступна лише невелика кількість позначених даних.

Таким чином для аналізу оптимальних архітектур та алгоритмів треба врахувати можливість ефективної роботи з послідовностями даних.

Як було визначено з аналізу існуючих дослідів, здебільшого у задачах визначення типу ФА людини використовують такі сенсори як: гіроскоп, акселерометр, gps-датчик, рідше магнітометр та мікрофон.

Беручи до уваги особливості предметної області фітнес-клубу та визначені раніше вимоги до підсистеми визначення типу ФА, актуальними джерелами даних для нас можу слугувати інформація з акселерометра та гіроскопа, зібраних з особистих пристроїв користувачів. GPS-датчик використовувати було б доцільно у випадку відстеження типу ФА людини на відкритому повітрі, тоді gps дані можна було б використовувати як джерело про швидкість руху відстежуваного об’єкту, магнітометр, можливо, для визначення особливостей рельєфу, але враховуючи що користувачі будуть знаходитися у фітнес-клубі, тобто певного приміщення з обмеженим простором, то користі від gps даних та магнітометру буде дуже

мало. Через високий рівень фонового шуму у приміщенні закладу: фонова музика, інші відвідувачі закладу, звуки обладнання, – мікрофон як стабільне джерело актуальної інформації також відпадає. Магнітометр загалом може давати актуальну інформацію яку можна використати під час класифікації, але слід враховувати що сам сенсор підлягає чималим перешкодам з боку особистих аксесуарів з магнітами, різного обладнання закладу з магнітними елементами тощо. Тобто дані з магнітометру мають чималий шанс виявитися вибросами, шо ускладнить сам процес проектування та розробку класифікатора для зменшення впливів цих вибросів на результат класифікації.

Тип використаних сенсорів також, здебільшого, визначає місце розташування сенсору на тілі людини. Розглядаючи описану предметну галузь, відвідувачі фітнес-клубу не будуть мати бажання кожного разу надягати спеціальне обладнання та проведення збору даних, то ж треба обмежитися персональними пристроями клієнтів, які можуть знаходитися на обмеженій кількості місць на тілі людини. Наприклад положення смартфона чи смарт-годинника відносно тіла людини сильно обмежено або кишенею або зап’ястям рук, то ж зібрані дані з особистих пристроїв різних людей будуть мати схожу семантику, що підвищить точність роботи класифікатора при загальному підході проектування системи.

Окрім визначених сенсорів також важливим моментом є частота дискретизації, або частота оновлення даних з цих сенсорів. Частота дискретизації – це кількість показань даних сенсора, записаних за одиницю часу. Частота дискретизації яку використовували в інших дослідженнях схожих задач по визначенню типу ФА варіюється від 10 до 100 Гц. Було також визначено що мінімальна оптимальна частота дискретизації становить від 20 Гц [6], але звісно чим вища частота дискретизації тим точніше буде результат роботи класифікатора, але треба враховувати що на підтримку високої частоти оновлення даних треба витрачати ресурси батареї пристроїв клієнтів, то ж питання визначення оптимальної частоти

дискретизації для задачі проектованої підсистеми стає одним з предметів дослідження даної роботи.

Враховуючи різноманіття алгоритмів та підходів до реалізації нейронних мереж з прямою кореляцією ефективності роботи моделі в залежності від виду вхідної інформації треба обрати певні архітектури та алгоритми які будуть ефективно працювати з даними сенсорів акселерометру та гіроскопу.

Нещодавні дослідження показують потенціал RNN (Recurrent neural network) при роботі з потоковими даними, і що особливо корисно саме з даними з акселерометра [10, 12]. Оскільки RNN була спроектована для вирішення основних проблем мереж прямого зв’язку, таких як:

- не здатність обробити послідовні дані;
- система враховує лише поточний вхід;
- неможливість запам'ятати попередні введення.

RNN, у свою чергу, може обробляти послідовні дані, приймаючи поточні вхідні дані та раніше отримані вхідні дані. RNN можуть запам'ятовувати попередні введення завдяки своїй внутрішній пам'яті. При цьому при реалізації внутрішньої пам’яті мережі можливі різні алгоритми що дає додаткову гнучкість при розробці та проектування системи.

1.5 Актуальність розглянутої проблеми

Розробка автоматичної системи для визначення типу ФА в спортивному залі чи фітнес-клубі стала надзвичайно важливою в останні роки. Така система може точно відстежувати та моніторити фізичну активність людей, надаючи цінні відомості про інтенсивність та тривалість їх тренувань. Ці дані можуть бути використані для налагодження тренувань та програм навчання, щоб задовольнити потреби кожного і поліпшити загальний досвід відвідування фітнес-клубу.

Для розробки ефективної системи виявлення фізичної активності у фітнес-клубі важливо дослідити можливі технології та методології виявлення та класифікації типу фізичної активності точно та ефективно. Для цього все більше використовуються алгоритми машинного навчання та глибокого навчання, оскільки вони можуть аналізувати великі кількості даних та робити точні прогнози на основі патернів та тенденцій, зафіксованих за певний період часу. Використання цих алгоритмів дозволяє створити інтелектуальну систему, яка може виявляти та розпізнавати різні типи фізичної активності.

Окрім автоматизації окремих бізнес-процесів фітнес клубу та надання більш повної статистики щодо власних тренувань користувачам, ефективна система виявлення фізичної активності у фітнес-клубі може мати також більш широкі наслідки для громадського здоров'я. Шляхом точного відстеження рівнів фізичної активності, фахівці з охорони здоров'я можуть отримати цінні відомості про патерни фізичної активності та сидячого способу життя, що може допомогти у розробці політик та ініціатив з громадського здоров'я. Це може призвести до розробки інтервенцій на основі спільнот, призначених для сприяння фізичній активності та зменшення ризику хронічних захворювань, таких як ожиріння та діабет.

Отже, дослідження та розробка системи для автоматичного визначення ФА людини у фітнес-клубі має потенціал змінити спортивну індустрію. Шляхом точного моніторингу та відстеження фізичної активності, ця система може надати цінні відомості, які можуть бути використані для поліпшення індивідуального досвіду клієнтів та підвищення загального здоров'я та благополуччя. З продовженням досліджень та розробок у цій галузі можливо створити систему, яка може бути корисною як для окремих осіб, так і для громад.

1.6 Постановка задачі

У рамках даної роботи необхідно провести дослідження можливих технологій, проектування та розробку системи визначення типу ФА людини. Визначити оптимальний метод розробки системи який надасть максимальну ефективність визначення типу ФА при найменших витратах ресурсів, як і апаратних так і обчислювальних.

Останні дослідження показали, що RNN мають значний потенціал для класифікації динамічних сигналів, зокрема для даних акселерометра, тому початковою точкою дослідження буде слугувати дослідження оптимальності використання різних типів RNN, а саме Long Short Term Memory (LSTM) та Gated Recurrent Units (GRU), адже саме RNN архітектура нейронної мережі найбільше відповідає структурі даних які обумовленні особливостями предметної галузі.

Крім того, буде проведено аналіз щодо оптимальних характеристик вхідних даних, способів препроцесінгу сирих даних та засобів оптимізації системи. Наприклад, можливими методами препроцесінгу можуть бути згладжування, фільтрація та нормалізація даних, а засобами оптимізації – відповідно використання різних архітектур мережі з метою визначення оптимальної.

У процесі проектування та розробки системи визначення типу ФА людини головна мета буде у забезпеченні максимальної точності при найменших витратах ресурсів, які включають як апаратні, так і обчислювальні витрати.

Як результат дослідження, ми надаємо порівняльну характеристику досліджуваних підходів до розробки системи визначення типу ФА людини, що дозволить визначити оптимальний метод розробки даної системи. У якості результат дослідження необхідно навести порівняльну характеристику досліджуваних підходів до розробки системи визначення типу ФА людини.

Для досягнення поставлених задач передбачається виконання наступних етапів:

– пошук та аналіз існуючої літератури у вигляді наукових статей та доповідей наукових журналів зі схожими задачами;

– визначення оптимальних архітектурних рішень на основі досліджених матеріалів, задля визначення найкращої альтернативи для вирішення задачі дослідницької роботи;

– детермінувати математичну модель досліджуваних алгоритмів;

– провести програмну реалізацію досліджуваних алгоритмів та підготувати метрики оцінювання ефективності роботи системи;

– провести аналіз вхідних даних системи та провести збір навчального та тестового набору даних;

– порівняти оцінки ефективності роботи розроблених систем між собою;

– провести аналіз отриманих результатів та визначити найкращу альтернативу з досліджуваних варіантів.

2 НЕЙРОМЕРЕЖЕВИЙ КЛАСИФІКАТОР

2.1 Архітектура класифікатора на основі RNN

Рекурентні нейронні мережі (RNN) є типом нейронних мереж глибокого навчання, спеціально розроблених для обробки послідовностей даних. Long short-term memory (LSTM) та gated recurrent units (GRU) є двома підтипами RNN, які були розроблені для обробки послідовних даних. Останні дослідження показали, що RNN мають значний потенціал для класифікації динамічних сигналів, зокрема для даних акселерометра.

Однак висока обчислювальна складність цих алгоритмів є значним викликом через велику кількість алгебраїчних операцій, які вони виконують. Наприклад, запуск цих моделей на пристроях з обмеженими обчислювальними ресурсами може призвести до довгого часу відповіді на запити, високого споживання енергії та нераціональних термінів виконання поставленої задачі. Це ускладнює процес проектування та реалізації систем визначення типу ФА людини у реальному часі на основі RNN.

Незважаючи на ці виклики, використовуючи усі доступні літературні та аналітичні ресурси треба визначити оптимальний варіант для реалізації даної системи з мінімізацією витрат на її функціонування.

Отже, RNN, на основі LSTM та GRU блоків, показали перспективні результати для класифікації динамічних сигналів, але їх велика обчислювальна складність ставить виклики для застосування в реальному часі. Але вони надають значну перевагу вирішуючи проблему зникаючого та вибухового градієнту [12], [13]. Запропоноване дослідження має на меті вирішити ці виклики та оцінити можливість реалізації системи визначення типу ФА людини на основі RNN.

Більш детально про кожний з підвидів рекурентних нейронних мереж.

Рекурентні нейронні мережі з керованими затворами (GRNN) – архітектура RNN, яка надає ефективне рішення проблеми зникаючого та

зростаючого градієнту, певним чином змінюючи алгоритм зворотного поширення помилки в часі, що призводить до більш ефективного процесу тренування. Центральна ідея за цією архітектурою – це комірка пам'яті з нелінійними механізмом затворів. Клітини пам'яті утримують інформацію ізольовано, зберігаючи свій стан протягом часу. Інформація керується за допомогою набору функцій активації, які називаються затворами. Під час процесу тренування для кожної комірки пам’яті налаштовуються ваги активації, тобто виконується навчається: закривати або відкривати свої затвори в залежності від вхідної інформації, отриманої з послідовності та інформації, яка зберігається в даний момент. Цю інформацію використовують у процесі навчання класичної RNN.

Використання GRNN стає все більш популярним в останні роки завдяки їх здатності ефективніше обробляти послідовні дані, ніж традиційні RNN. Фактично, вони успішно застосовуються в різних галузях, таких як обробка природної мови, визнавання зображень і мовлення, виявлення аномалій та інших. Крім того, архітектура GRNN надає можливість проводити більш детальний контроль над потоком інформації, що може призвести до кращих результатів в завданнях, які потребують довгострокових спостережень та ітераційних корегувань.

Комірки довго коротко-строкової пам’яті (LSTM) були першими з запропонованих gated RNN підходів. Вони мають три затвори, два з яких – вхідний затвор та затвор забуття – визначають, чи додавати нову інформацію до пам'яті, або видаляти частину збереженої інформації, відповідно. Третій затвор, який називається вихідним затвором, контролює, яка інформація передається наступному рівню нейронної мережі в процесі навчання. Математичну модель яка описує LSTM RNN, можна виразити як набір векторів (2.1 – 2.6).

$$e_t = o^t \circ tahn(c^t), \tag{2.1}$$

$$o^t = \sigma(W_{xo}x^t + W_{ho}h^{t-1} + w_{co} \circ c^t + b_0), \quad (2.2)$$

$$c^t = f^t \circ c^{t-1} + i^t \circ \tilde{c}^t, \quad (2.3)$$

$$\tilde{c}^t = tahn(W_{xc}x^t + W_{hc}h^{t-1} + b_c), \quad (2.4)$$

$$f^t = \sigma\left(W_{xf}x^t + W_{hf}h^{t-1} + w_{cf} \circ c^{t-1} + b_f\right), \quad (2.5)$$

$$i^t = \sigma(W_{xi}x^t + W_{hi}h^{t-1} + w_{ci} \circ c^{t-1} + b_i), \quad (2.6)$$

де h^t це одиничний стан, c^t представляє пам'ять комірки, коли $\tilde{c}^t$ представляє нову інформацію яка приходе з самої мережі. У свою чергу вектори i^t, o^t, f^t це вихідна інформація затворів комірки, вхідного затвору, вихідного та затвору забування відповідно. σ представляє собою сигмоїдну, а $tahn$ гіперболічну функцію активації. Векторне по-елементне множення позначене символом «∘».

У описі векторів (2.1 – 2.6) описані також наступні ваги:

– вхідні ваги: $W_{xf}, W_{xo}, W_{xc}, W_{xi} \in R^{N*M}$;

– рекурентні ваги: $W_{hf}, W_{ho}, W_{hc}, W_{hi} \in R^{N*M}$;

– ваги комірок: $w_{cf}, w_{co}, w_{ci} \in R^N$;

– ваги зміщення: $b_f, b_o, b_i, b_c \in R^N$.

Де N – кількість блоків LSTM, а M – кількість входів.

Графічне представлення блоку LSTM наведено на рисунку 2.1.

З іншого боку, керовані рекурентні блоки (GRU) є найновішою альтернативою LSTM для реалізації комірки пам'яті. Основна відмінність полягає у відсутності вихідних затворів, тому те, що зберігається в пам'яті комірки, повністю передається в мережу протягом усього процесу навчання. Інші затвори виконують функцію очищення даних, збережених з попередніх ітерацій, та збереження даних які приходять у вигляді вхідної інформації.

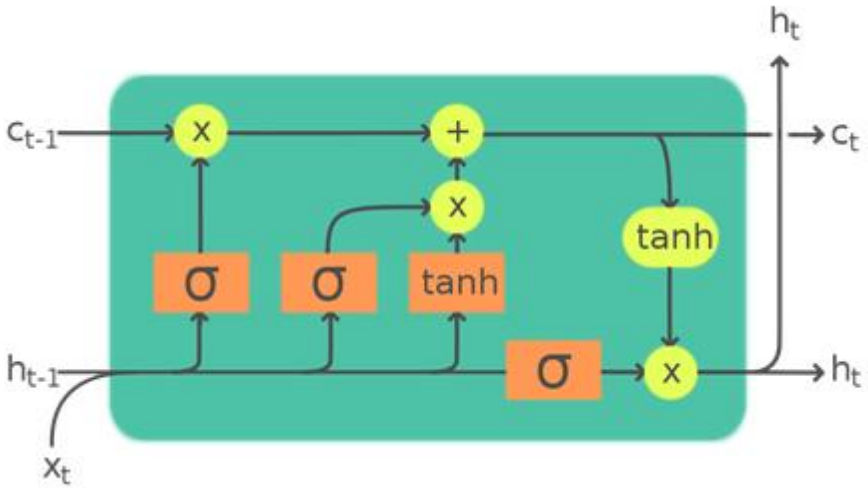

Рисунок 2.1 – Графічне представлення LSTM блоку

Рівняння GRU затворів досить відрізняються від тих, що моделюють LSTM, переважно через відсутність вихідних затворів (2.7 – 2.10).

$$h^t = (1 - z^t) \circ h^{t-1} + z^t \circ \tilde{h}^t, \tag{2.7}$$

$$z^t = \sigma\left(W_{xf} x^t + W_{hz} h^{t-1} + b_z\right) \tag{2.8}$$

$$\tilde{h}^t = tahn(W_{xc} x^t + W_{hc}(r^t \circ h^{t-1})), \tag{2.9}$$

$$r^t = \sigma(W_{xr} x^t + W_{hr} h^{t-1} + b_r), \tag{2.10}$$

де z^t та r^t це результат затвори оновлення та забування відповідно. У описаних векторах (2.7 – 2.10) використовується дещо менша кількість ваг, серед яких:

– вхідні ваги: $W_{xz}, W_{xc}, W_{xr} \in R^{N*M}$;

– рекурентні ваги: $W_{hz}, W_{hc}, W_{hr} \in R^{N*M}$;

– ваги зміщення: $b_z, b_r \in R^N$.

Графічне представлення GRU блоку наведене на рисунку 2.2.

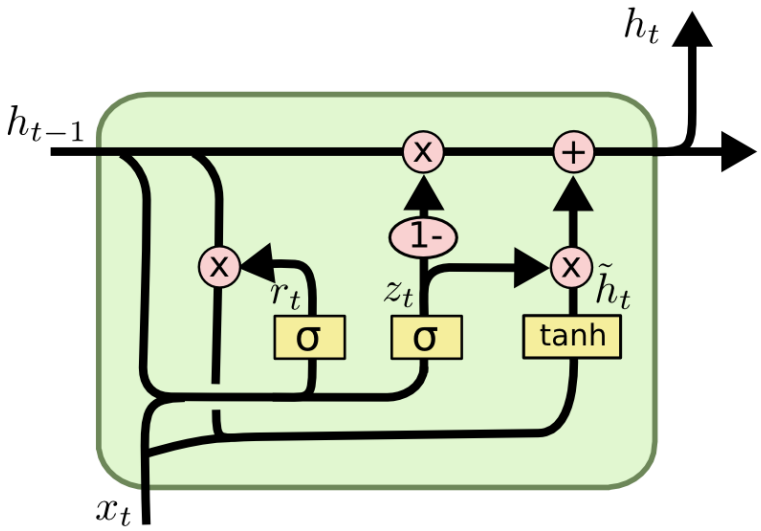

Рисунок 2.2 – Графічне представлення GRU блоку

Обидві розглянуті альтернативи показують схожу ефективність при виконанні загальних задач, але підхід з використання керованих рекурентних блоків вимагає дещо менше обчислювальних ресурсів через зменшену кількість ваг та векторів.

Загально прийняті методи до проектування нейромережевих систем та окремі результати досліджень показують, що нормалізація значень вибірки значно покращує ефективність, адже таким чином ми зменшуємо потенційне перетренування системи. Для досягнення цього, на початку архітектури додається шар нормалізації пакету вхідних даних. Однак, дослідження з 10-кратною перехресною перевіркою [14] показало, що цей підхід не є ефективним для отримання не послідовних характеристик.

Спираючись на певні розбіжності у можливих засобах реалізації нормалізації вхідних даних, у рамках даного дослідження буде проведено додаткову перевірку різних архітектур мережі, для визначення якомога точного підходу до реалізації системи під наші цілі.

Будемо розглядати чотири архітектури, які складаються з двох найпростіших моделей з нормалізацією пакету, рівнем RNN та вихідних рівнем. У якості функції активації вихідного рівня для визначення класу

екземпляру вхідних даних буде слугувати Softmax. Основною відмінністю між цими двома архітектурами полягає в тому, що в якості рекурентного шару використовується LSTM або GRU. Інші дві архітектури містять додатковий другий шар RNN того ж типу, що й попередній. Хоча обчислювальні витрати більш складних версій вищі, це потенційно може призвести до відчутного приросту ефективності роботи мережі, що у свою чергу може бути виправдано вимогами окремого бізнесу.

Таким чином, як вказано на рисунку 2.3 та 2.4, у якості dense шарів системи буде використано або один або два шари RNN, реалізованих з використання LSTM чи GRU комірок пам'яті, у комбінації з фінальним dense рівнем з простою семантикою. Таким чином, провівши порівняльний аналіз результатів ефективності роботи системи, можна буде визначити оптимальну архітектуру мережі.

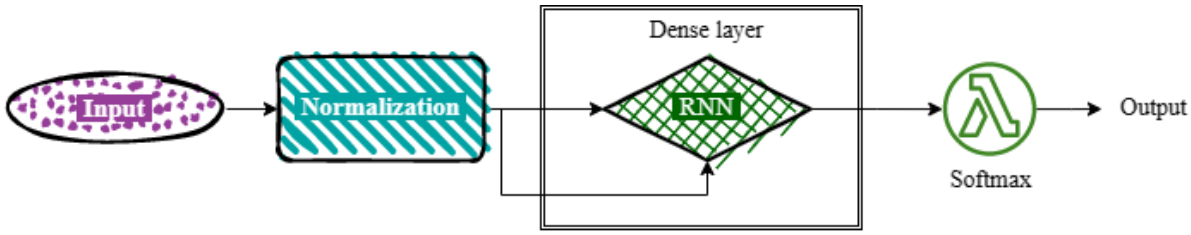

Рисунок 2.3 – Архітектура мережі з одним шаром RNN

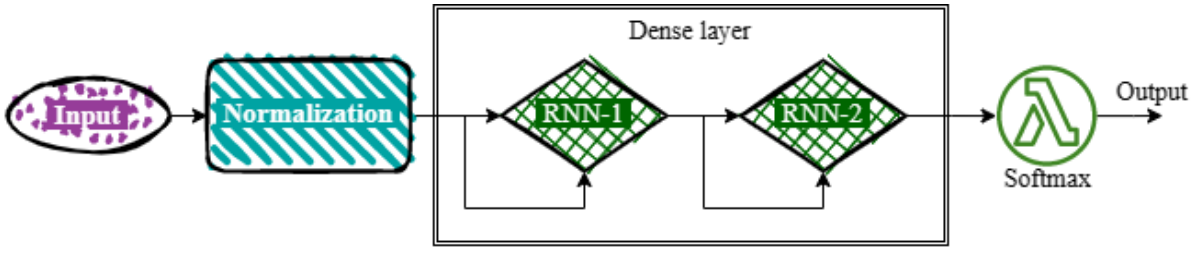

Рисунок 2.4 – Архітектура мережі з двома шарами RNN

Один з поширених патернів проектування нейронних мереж полягає в використанні dense шару з функцією активації Rectified Linear Unit (ReLu)

як останнього шару. Цей патерн часто використовується в задачах класифікації, де мережа навчається передбачати до якої з кількох класів належить вхідний сигнал. Функція активації ReLu відома своєю здатністю покращувати швидкість та стабільність процесу навчання, що робить її популярним вибором для багатьох архітектур нейронних мереж. Крім того, цей фінальний dense шар дозволяє мережі комбінувати інформацію з усіх попередніх шарів та приймати остаточне рішення на основі цієї комбінованої інформації.

2.2 Опис апаратних складових при проектуванні та розробці системи

Під час проектування та розробки системи визначення типу ФА людини було використано наступні апаратні засоби для навчання та виконання тестів системи:

– у якості особистого пристрою при імітації діяльності клієнта фітнес-клубу було використано смартфон Apple iPhone 12 mini та смарт-годинник Apple Watch Series 7;

– у якості машини для розробки та навчання системи було використано комп’ютер Apple Mac mini з вмонтованим процесором Apple Silicon M2.

Слід зазначити що використані апаратні пристрої мають певну специфічну апаратну складову у вигляді окремих нейронних обчислювальних компонентів, які представляють з себе спеціалізовані обчислювальні пристрої, які призначені для прискорення завдань машинного навчання, таких як розпізнавання зображень, обробка природньої мови та розпізнавання мовлення. У наведених особистих клієнтських пристроях використовуються процесори A15 Bionic для смартфону та S7 для смарт-годинника, у яких є окремий підмодуль процесору який спеціально розроблений для задач машинного навчання. A15 Bionic має у своєму складі 16-ядерний нейронний підпроцесор, який

може виконувати до 15,8 трильйонів операцій на секунду окремо від центрального процесору, тоді як чіп S7 має 16-ядерний підпроцесор, яка може виконувати до 11 трильйонів операцій на секунду.

Окремі нейронні підпроцесори в цих чіпах забезпечують кілька переваг. По-перше, вони дозволяють швидше та ефективніше обробляти задачі машинного навчання, що може покращити продуктивність додатків, які ґрунтуються чи використовують ці технології. По-друге, вони зменшують навантаження на центральний та графічний процесор, що може призвести до кращої тривалості роботи батареї та меншого вироблення тепла. По-третє, вони дають можливість виконання певних задач машинного навчання на самому пристрої, які раніше не були можливі на мобільних пристроях без делегації обчислювальних операцій на більше спеціалізоване обладнання методами інтернету.

Крім підпроцесорів, які входять до складу цих чіпів, A15 Bionic та S7 включають інші спеціалізовані апаратні компоненти, які призначені для покращення продуктивності в конкретних областях. Наприклад, чип A15 Bionic включає новий ISP – препроцесінг зображення при зйомці на смартфон.

В цілому, нейронні підпроцесори та інші спеціалізовані апаратні компоненти в цих чіпах представляють значний технологічний прогрес в галузі мобільної обробки. Вони забезпечують швидше та ефективніше обробляти задачі машинного навчання, що може покращити продуктивність додатків та дозволяти нові типи додатків. Таким чином вирішення даної задачі буде актуально перевірити виконавши всі задачі по класифікації саме на особистому пристрої клієнта закладу, для забезпечення високого рівня ізоляції програмного додатку фітнес-клубу від якості інтернет з’єднання та наявності інших допоміжних ресурсів.

2.3 Аналіз порівняльних метрик проектованої системи

Процес навчання проектованих систем планується проводити за власноруч зібраними показниками сенсорів зібраними для однієї людини. Для забезпечення оптимального засобу перевірки ефективності системи для навчання і тестування системи збір даних буде виконано декілька разів у різні періоди часу. Умовно збір даних для тренування буде проводитися впродовж двох неділь, для тестування дані будуть збиратися впродовж третьої неділі, для мінімізації ідентичності даних при тестуванні системи.

Для проведення дослідження у гарних умовах треба мати вибірку даних для більшої кількості осіб, і використовувати частину даних однієї групи людей для навчання моделей, іншу частину людей використовувати для тестування системи. У даному випадку дослідження має більш індивідуальний характер, тож отримані результати дослідження можуть відрізнятися у випадку дослідження та розробки багато-користувацької системи.

Враховуючи індивідуальні особливості ФА окремої людини – певні пороги гнучкості окремих кінцівок при виконанні певних вправ, чи фізіологічні особливості, чим більша вибірка людей буде брати участь при навчання системи тим потенційно менша ефективність буде у системи. Таким чином для проектованої системи варто розглянути додатковий модуль керованого навчання для забезпечення більшої індивідуальності під час класифікації ФА людини, але це буде розглядатися у наступних роботах.

Під час виконання етапу навчання та тестування системи буде проведено п’ять етапів навчання та тестування кожної з аналізованих систем. У кожній ітерації навчання і тестування будуть використані різні частоти дискретизації сигналу для визначення оптимальної частоти для мінімізації використання апаратних ресурсів під час роботи системи зі збереженням адекватної ефективності класифікації.

Після визначення оптимальної частоти буде проведено оптимізацію архітектури системи шляхом оптимізації значень гіперпараметрів моделі. Гіперпараметри – це параметри моделі, які не вивчаються перед навчанням, які будуть безпосередньо впливати на процес навчання системи. Приклади гіперпараметрів: швидкість навчання, розмір пакета і кількість прихованих шарів чи нейронів у кожному шарі нейронної мережі.

Сама оптимізація буде проводитися методом grid search. Для виконання grid search визначається діапазон значень для кожного гіперпараметра. Потім модель навчається та оцінюється з кожною можливою комбінацією гіперпараметрів в межах визначених діапазонів. Комбінація гіперпараметрів, яка дає найкращу ефективність на тестовому наборі вхідних даних, вибирається як оптимальний набір гіперпараметрів.

Хоча grid search може бути ефективним способом знаходження оптимальних гіперпараметрів, він також може бути достатньо витратним з точки зори кількості обчислень, особливо для моделей з багатьма гіперпараметрами або великими діапазонами можливих значень. У результаті, інші техніки, такі як випадковий пошук та байєсівська оптимізація, також часто використовуються для пошуку найкращих гіперпараметрів.

Враховуючи високий дисбаланс датасету, в основному через природи вхідних даних, звичайний підхід до оцінки ефективності моделі у вигляді загальної точності класифікації не є раціональним для аналізу. Неоднорідність даних зумовлена тим фактом що певні ФА людина може виконувати та, як правило, виконує довго – наприклад біг чи інші кардіо вправи, у той час як силові вправи зосереджені на циклічні рухи тіла з певною періодичністю у обмежений термін. Тому для оцінки ефективності роботи системи буде використано метрику MF1 [15]. Вона розраховується шляхом аналізу гармонійного середнього значення точності та повторюваності для кожного класу в наборі даних, з врахуванням середнього значення цих оцінок для всіх класів (2.11 – 2.14). Ця метрика

надає однакову вагу кожному класу, незалежно від кількості зразків у кожному класі. Високий MF1 свідчить про те, що модель працює добре для всіх класів, тоді як низька оцінка свідчить про те, що модель має проблеми з точною класифікацією певних класів.

$$Macro\ F1\ Score = \frac{\sum_{i=1}^{n} F1Score_i}{n}, \tag{2.11}$$

$$F1Score = \frac{2*Precision*Recall}{Precision+Recall}, \tag{2.12}$$

$$Precision = \frac{TP}{TP+FP}, \tag{2.13}$$

$$Recall = \frac{TP}{TP+FN}, \tag{2.14}$$

де n – кількість класів у системі, а TP, FP, FN відображають кількість позитивно правильних результатів, кількість негативно правильних результатів, та повністю неправильних результатів класифікації екземпляру класу відповідно.

Відповідно до аналізу ефективності моделей класифікації також використовують такі метрики як чутливість та специфічність [14]. Чутливість – це метрика, яка вимірює здатність моделі машинного навчання виявляти позитивні екземпляри і також відома як дійсний позитивний рейтинг або повторність. Оцінюючи продуктивність моделі на основі чутливості, ми можемо визначити кількість позитивних екземплярів, які модель правильно визначила. Щоб проілюструвати цю концепцію, розглянемо приклад медичного тесту на рідкісну хворобу. Якщо тест має чутливість 95%, це означає, що зі ста людей, які мають цю хворобу і проходять тест, 95 з них будуть правильно визначені як позитивні, але 5 – неправильно (хибні негативи).

$$Sensitivity = Recall = \frac{TP}{TP+FN}. \quad (2.15)$$

Специфічність відповідає відсотку правильно ідентифікованих негативних результатів. Це означає, що є певний відсоток дійсних негативів, які визначаються системою як позитивні – хибні позитиви. Цей відсоток також може бути відомий як рівень правильних негативів. Важливо зауважити, що сума специфічності (рівня правильних негативів) та рівня хибних позитивів завжди дорівнює 1. Висока специфічність означає, що модель точно ідентифікує більшість негативних результатів, тоді як низька специфічність означає, що модель неправильно класифікує багато негативних результатів як позитивні.

$$Specificity = \frac{TN}{TN+FP}, \quad (2.16)$$

де TN – кількість правильних негативів, а FP – кількість негативно правильних результатів.

3 ЗБІР ТА АНАЛІЗ ВХІДНИХ ДАНИХ

3.1 Процес збору показників сенсорів

Для збору даних було використано спеціально розроблений додаток для телефону для смарт-годинника на платформі iOS, за допомогою якого була забезпечена можливість знімати показники сенсорів акселерометру, гіроскопу та стану батареї пристрою для забезпечення аналітичною та змістовною інформацією, необхідної для порівняльного аналізу методів реалізації.

У додатку є можливість налаштувати частоту дискретизації сигналу, який буде записаний до окремого файлу, та є можливість промаркувати окремий файл умовним позначенням у чисельному форматі [9], для облегшення препроцесінгу сирих даних. Особливістю даної системи та даних необхідних для навчання мережі є необхідність розгляду лише певного відрізку часу, коли користувач безпосередньо виконує певну фізичну вправу. Таким чином при читанні показників з сенсорів кожен підхід був записаний до окремого файлу. Виходячи з особливостей запису показників, певний відрізок часу на початку та вкінці не матиме цінності при навчання, адже тоді виконувалися підготовчі дії для логування. У рамках даних дослідів, виключення цих підготовчих інтервалів логування кожного підходу призведе до зосередженні системи на детермінації саме акту виконання дії, а не факт початку виконання, адже у дослідних умовах зімітувати природній акт ініціації виконання окремої ФА є дуже вибагливою задачею, та і під час аналізу ПО не було виявлено бізнес-функцій які б були автоматизовані при наявності даного функціоналу, тому цю задачу було відкинуто.

Таким чином при виконання препроцесінгу та нормалізації вхідних даних необхідно врахувати необхідність тримування початкових та кінцевих записів сирих даних.

Як можна спостерігати на рисунку 3.1, на початку спостереження окремої активності, у даному випадку це тяга тросу до поясу сидячи (рисунок А.5), на початку файлу логування перші приблизно 350 записів при частоті 100 ГЦ не співпадають з основним патерном рухів сенсору, саме цей часовий проміжок необхідно обрізати, як на початку так і в кінці кожного файлу з показниками сенсорів.

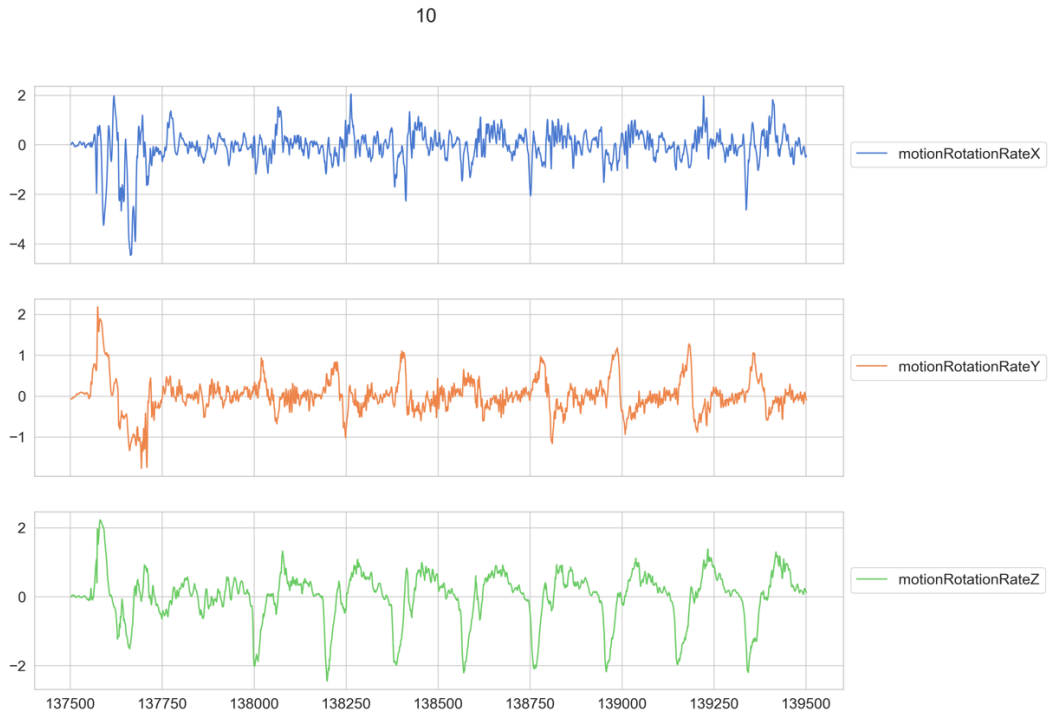

Рисунок 3.1 – показники гіроскопу при виконанні вправ на тренажері-греблі

Для аналізу мінімально оптимальної частоти дискретизації сигналу з сенсорів, та аналізу витрат акумуляторів особистих пристроїв, до переліку даних з просторових сенсорів також було виведено показники значення заряду акумулятора пристрою, для визначення динаміки витрачення заряду при окремих частот дискретизації.

Слід зазначити що для аналізу ефективності мережі з різними частотами дискретизації даних було використано дещо інший підхід. Сирі дані були зібрані у одному екземплярі з максимально можливою частотою

дискретизації, і надалі, для імітації роботи сенсорів з різними частотами, частота буде штучно зменшена шляхом селективного вибору сигналів з максимально повного набору даних. Таким чином ми зменшимо необхідні витрати на збір корисних даних у тренажерному залі [14].

У свою чергу для визначення витрат заряду батареї при читанні показників сенсорів на різних частотах буде проведено окремий дослід, при виконанні довільних ФА на довгому проміжку часу, для визначення довгострокової динаміки витрат апаратних ресурсів.

І в результаті аналізу результатів двох різних дослідів ми будемо мати оптимальний набір частоти дискретизації сирих даних з точку зору ефективності класифікатора та рівню споживання заряду акумулятора особистих пристроїв.

3.2 Апаратні витрати засобу збору даних

Відповідно до окремого тестування апаратних витрат на збір та запис показників сенсорів з особистого пристрою, а саме Apple Watch Series 7, було визначено, що за годину запису показників з частотою дискретизації сигналу 100 ГЦ, було витрачено 1 процент заряду акумулятору пристрою.

Відповідно до загальних технічних характеристик комбінованого сенсору акселерометр-гіроскоп, середнє споживання електроенергії становить 0.2 мА/год, таким чином враховуючи ємність акумулятору 309 мА/год робота комбінованого сенсору не буде великою статтею витрат заряду, навіть з урахуванням особливостей окремих операційних систем які перехоплюють показники сенсорів для їх попередньої обробки та формування можливого до обробки формату.

Враховуючи предметну область та особливості бізнес-процесів фітнес клубу, можна врахувати можливість створення власного апаратного забезпечення у вигляді окремого браслету з прикріпленим до нього сенсором акселерометр-гіроскоп, який буде записувати показники у

локальну флеш пам’ять, після чого дані будуть передані на окремий сервер де буде проведено аналіз тренування користувача з визначенням часових інтервалів окремих тренувань. Для реалізації подібного проекту буде достатньо використовувати будь-який комбінований сенсор з мінімальною частотою оновлення не менше 25 ГЦ [6], та з можливістю знімати трьох вимірні показники прискорення та положення сенсора. На даний момент на ринку в наявності чимало альтернатив подібних сенсорів.

3.3 Препроцесінг та нормалізація сирих даних

Серед особливостей розробленої програми для збору та запису показників сенсорів годинника та телефону є можливість виведення нормалізованих значень відносно різних природних умов. Наприклад використаний пристрій, а саме Apple Watch Series 7 має змогу надавати показники акселерометра та гіроскопа з урахуванням факторів гравітації, тобто пристрій автоматично відфільтровує рухи та зміни у положенні які не були ініційовані самим користувачем. У випадку використання інших апаратних засобів для збору показників з сенсорів треба врахувати апаратну можливість пристрою, або окремо дослідити можливість програмної нормалізації значень відносно природних факторів.

Інтерфейс розробки програмних засобів для портативних пристроїв компанії Apple надає первинну логіку по визначенню типу активності людини у реальному часу. Але, у результаті спостереження за показниками відповідних полів, було визначено що пристрій автоматично має змогу визначати лише ступінь інтенсивності рутинних процесів, серед яких: покій, ходьба, інтенсивне і невідоме. Відповідно судячи зі спостережень при виконанні будь якої конкретної вправи пристрій ідентифікував активність як невідому, тому даний інтерфейс не відповідає вимогам до проектованої системи. На відміну від доступного інтерфейсу операційної системи, сам пристрій має попередньо встановлений додаток Fitness, але він не надає

розробникам та користувачам відкритого інтерфейсу до відстеження рухів користувача у реальному часі, хоча він і має значно більше покриття різних типів ФА. Але для проектованої системи необхідно забезпечити повний цикл відстеження початку тренування, самого тренування і факт його завершення, то ж використання додатку Fitness не дасть бажаного результату, адже для коректного функціонування логування необхідний активний зв’язок з користувачем, кількість якого необхідно мінімізувати у рамках даного проекту.

Окрім неадекватних даних на крайніх позиціях кожного файлу спостереження, сирі дані акселерометра та гіроскопа містять шум та іншу зайву інформацію, яка може вплинути на точність системи. Цей шум може виникати з різних джерел, таких як помилки датчиків, електричні перешкоди та механічні вібрації. Тому, для зменшення впливу шуму та отримання корисної інформації з сирого потоку даних, потрібно застосовувати фільтрацію різних частот.

Одним з типів фільтрів, які можна використовувати, є фільтр високих частот, який використовується для видалення шумів низької частоти та дрейфу даних. Цей тип фільтра особливо корисний для видалення шумів, які викликані гравітацією або власним рухом датчика. Цей фільтр, у нашому випадку, автоматично застосовується особистим пристроєм. З іншого боку, фільтр низьких частот може використовуватися для видалення шумів високої частоти, таких як швидкі випадкові рухи, які у більшості випадків не повинні впливати на результат класифікації відрізку часу з цим артефактом, чи вібрації викликаних окремими обладнанням для тренування чи іншими факторами.

Серед фільтрів низьких частот найбільш поширеним є фільтр Баттерворта (Butterworth), який використовується в цифровій обробці сигналів для видалення шумів високої частоти з послідовних даних. Це тип фільтру з безкінечною імпульсною характеристикою (IIR), який має

максимально плоску частотну характеристику в смузі пропускання та монотонне затухання в смузі затримки.

Однією з переваг фільтру Баттерворта є те, що він надає мінімальні спотворення сигналу в смузі пропускання. Однак, важливо зазначити, що фільтр не є ідеальним для всіх типів послідовних даних, оскільки він може викликати фазові спотворення та породжувати перешкоди в смузі пропускання.

Іншою альтернативою фільтру Баттерворта може слугувати фільтр Калмана. Фільтр Калмана є потужним математичним алгоритмом, який може використовуватися для оцінювання стану системи на основі серії вимірювань. Він працює, враховуючи як вимірювання, так і модель системи, за допомогою серії обчислень для корекції оцінки стану системи на основі нових вимірювань.

Однією з головних переваг фільтра Калмана є його здатність працювати з шумними або неповними даними. Це особливо корисно в разі даних акселерометра та гіроскопа, які можуть бути піддані різноманітним джерелам шуму та втручання. Застосовуючи фільтр Калмана до даних, ми можемо згладити шум та видалити будь-які викиди або раптові викиди в даних, що призведе до отримання більш точного та адекватного датасету.

У випадку даних акселерометра Калманівський фільтр може бути використаний для видалення будь-якої дрейфу або спотворення в читаннях, що можуть виникнути через фактори, такі як зміни температури або помилки калібрування датчика. Фільтр також може бути використаний для корекції будь-яких нелінійностей відповіді сенсора, що може призводити до неточних читань.

Аналогічно, фільтр може бути використаний для покращення точності даних гіроскопа, які можуть бути піддані впливу таких факторів, як шум сенсора, вібрація та дрейф. Застосовуючи фільтр до даних, ми можемо видалити ці джерела помилок та отримати більш точні показники швидкості обертання пристрою.

В цілому, фільтр Калмана є потужним інструментом для покращення точності даних акселерометра та гіроскопа. Завдяки використанню комбінації вимірювань та моделі системи, фільтр може продукувати більш точну інформацію про динаміку положення особистого пристрою у просторі, навіть в присутності шумних або неповних даних.

Загалом, застосовуючи комбінації фільтрів високих та низьких частот можна видалити більшість шумів з даних, надаючи більш точне представлення руху пристрою під час збору даних чи при виконанні саме класифікації.

Крім зменшення шуму, фільтри також можуть використовуватися для видобування конкретних ознак з даних, які є важливими для виявлення та класифікації рухів людини. Наприклад, фільтри можуть використовуватися для ізоляції конкретних діапазонів частот, які пов'язані з певними типами руху. Це може допомогти системі розрізняти різні за інтенсивністю типи діяльності, але враховуючи обрану архітектуру мережі, визначення окремих ознак даних не є необхідною задачею, хоча у випадку додаткового виявлення залежностей за певними правилами та доповнення показників сенсорів цими ознаками може призвести до збільшення точності системи, особливо у окремих суміжних за окремими сенсорами активностями.

В цілому, використання фільтрів є невід'ємною складовою для покращення точності та надійності систем виявлення руху людини. Видаляючи шум та видобуваючи відповідні ознаки з сирого потоку даних, система може точно виявляти та класифікувати рухи людини, що може бути корисним у різних застосуваннях, таких як відстеження фізичного стану, аналіз спортивної продуктивності та реабілітація.

Датасет був зібраних однією особою з використанням власноруч розробленого додатка на смарт-годинник та смартфон. Для збору даних було виставлено частоту дискретизації сигналу на рівні 100 Гц – це максимальна допустима частота яка доступна користувачу пристрою. Для подальших експериментів залежності ефективності системи від частоти

дискретизації вхідних даних, частота буде штучно зменшена шляхом залишення кожного другого (для 50 Гц) або кожного четвертого (для 25 Гц) запису. Загальна кількість записів при максимальній частоті становить 378011, і включає до себе 9 різних вправ, а саме: гребля (рисунок А.1), жим гантелей сидячи (рисунок А.2), тяга штанги до поясу (рисунок А.3), жим сидячи (рисунок А.4), підтягування, ходьба, біг, тяга тросу до поясу сидячи (рисунок А.5), розведення гантелей у бік (рисунок А.6). У свою чергу кількісний розподіл записів серед цих дев’яти активностей наведений на рисунку 3.2.

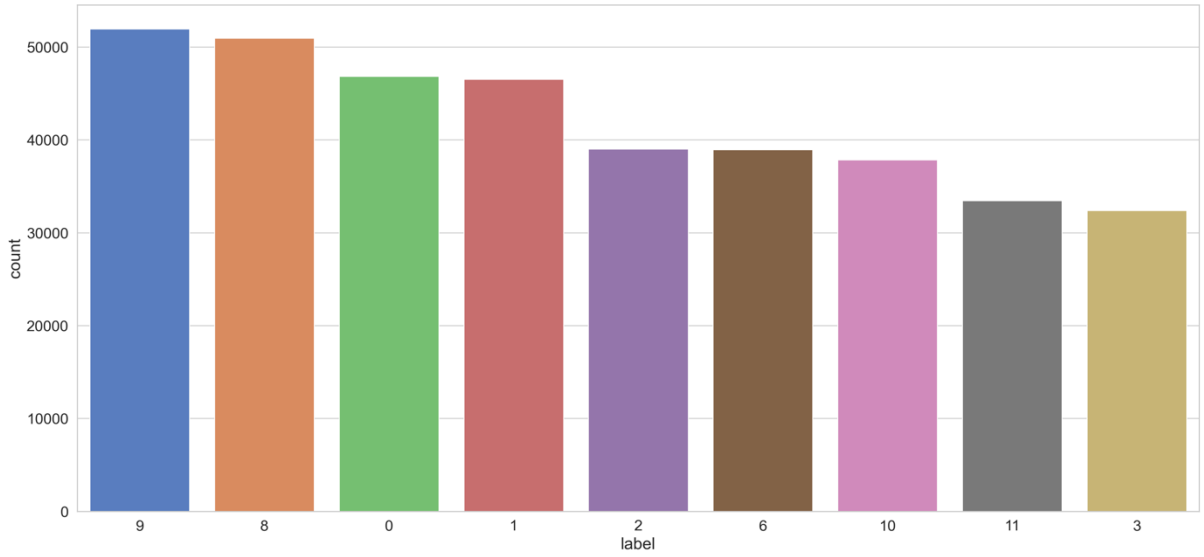

Рисунок 3.2 – Розподіл датасету на види ФА

4 АНАЛІЗ ПРИЙНЯТИХ ПРОГРАМНИХ ТА АРХІТЕКТУРНИХ РІШЕНЬ

4.1 Визначення глобальних параметрів системи

Під час проектування та розробки нейромережевого класифікатора треба враховувати чималу кількість параметрів системи які будуть впливати на процес навчання системи та на процес виконання саме класифікації. Такі параметри зазвичай називають глобальними гіперпараметрами системи, адже при зміні одного з параметрів можна отримати дуже різні результати ефективності і точності системи. Окремою задачею при розробці нейромережевих засобів автоматизації є процес визначення та оптимізації значень глобальний параметрів. Враховуючи високу складність системи з точки зору розуміння та обчислюваних операцій необхідних при розробці та аналізу, процес автоматизації глобальних параметрів системи є дуже складною задачею. Одним з популярних методів для оптимізації глобальних гіперпараметрів нейронних мереж є виконання grid search. Це передбачає тренування мережі з різними комбінаціями гіперпараметрів та вибір комбінації, яка дає найкращу продуктивність на валідаційному наборі.

Для виконання grid search спочатку ми визначаємо набір кандидатів на кожний гіперпараметр. Потім ми навчаємо мережу, використовуючи кожну можливу комбінацію гіперпараметрів у нашому наборі кандидатів. Після навчання ми оцінюємо продуктивність кожної моделі на валідаційному наборі та вибираємо набір гіперпараметрів, який дав найвищу продуктивність.

Grid search може бути розрахунково витратним процесом, оскільки він вимагає навчання та оцінки великої кількості моделей. Однак, це може бути потужним інструментом для налаштування продуктивності нейронних мереж та оптимізації їх гіперпараметрів для конкретного завдання, з

урахуванням альтернативи у вигляді розробки окремого додатку для оптимізації значень цих параметрів.

Для початку виконання пошуку оптимальних архітектурних рішень визначимо перелік параметрів які можуть приймати довільні значення. Серед таких можна визначити: по-перше характеристики сегментів даних, отриманих віконним методом розподілу вхідних даних, у вигляді довжини сегменту та рівня перехресності сегментів. По-друге, можна виділити кількість нейронів на окремих шарах мережі. По-третє, серед параметрів навчання системи можна виділити шаг навчання (learning rate), розмір пакету навчання (batch size), рівень забування (dropout rate). Таким чином визначивши оптимальні значення всіх гіперпараметрів ми отримаємо найефективнішу мережу для виконання поставленої задачі визначення типу ФА.

На початку навчання нейромережевого класифікатора треба провести підготовку сирих даних у вигляді препроцесінгу, до якого можна віднести нормалізацію, та сегментацію вхідних даних. Під час сегментації треба врахувати оптимальні інтервали на які будуть поділені усі активності під час навчання системи, які у подальшому будуть характеризувати мінімальні інтервали показників сенсорів для виконання класифікації. Розглядаючи архітектуру RNN зрозуміло, що для підвищення ефективності навчання системи та для мінімізації можливих артефактів моделі при навчанні, вхідні дані системи буде розділено на велику кількість однаково важливих інтервалів. Таким чином ми зменшимо вплив вибросів у показниках сенсорів на результат роботи класифікатора, адже система, яка отримала на вхід лише декілька інтервалів з артефактами, має більше шансів забути ці неадекватні патерни у послідовності даних, якщо у подальшому навчанні буде отримано тисячі адекватних інтервалів.

Але слід визначити оптимальні характеристики сегментів, серед яких можна виділити саме розмір одного сегменту і рівень перекривання одного сегменту іншим. Розподіл сигналу на незалежні сегменти є самим простим

варіантом сегментування даних, але він має певні концептуальні недоліки. Таким чином поділивши дані на незалежні сегменти ми потенційно втрачаємо важливі перехідні патерни, які можуть лежати по краях окремих сегментів, при цьому втрачаючи свою цілісність. У свою чергу розділення на незалежні максимально короткі сегменти, для мінімізації ризику втрати таких патернів, не є можливим через семантику показників розглянутих у цьому дослідженні. Річ у тім, що система має приймати у кожному сегменті мінімум одне повне повторення окремої фізичної вправи, для фіксування унікальних патернів кожної ФА, а через достатньо високий рівень різноманіття вправ за інтенсивністю (одне повторення жиму лежачи, наприклад, може зайняти до 3 секунд, а середній період при швидкому бігу може становити до 1 секунди), тривалість сегменту треба встановлювати відповідно до мінімально інтенсивного – чим довше відбувається одне повторення, тим більше повинен бути сегмент. Проаналізувавши графіки показників декількох вправ було визначено що одне повторення відбувається за середній максимум 2.5 секунди, як наведено на рисунку 4.1.

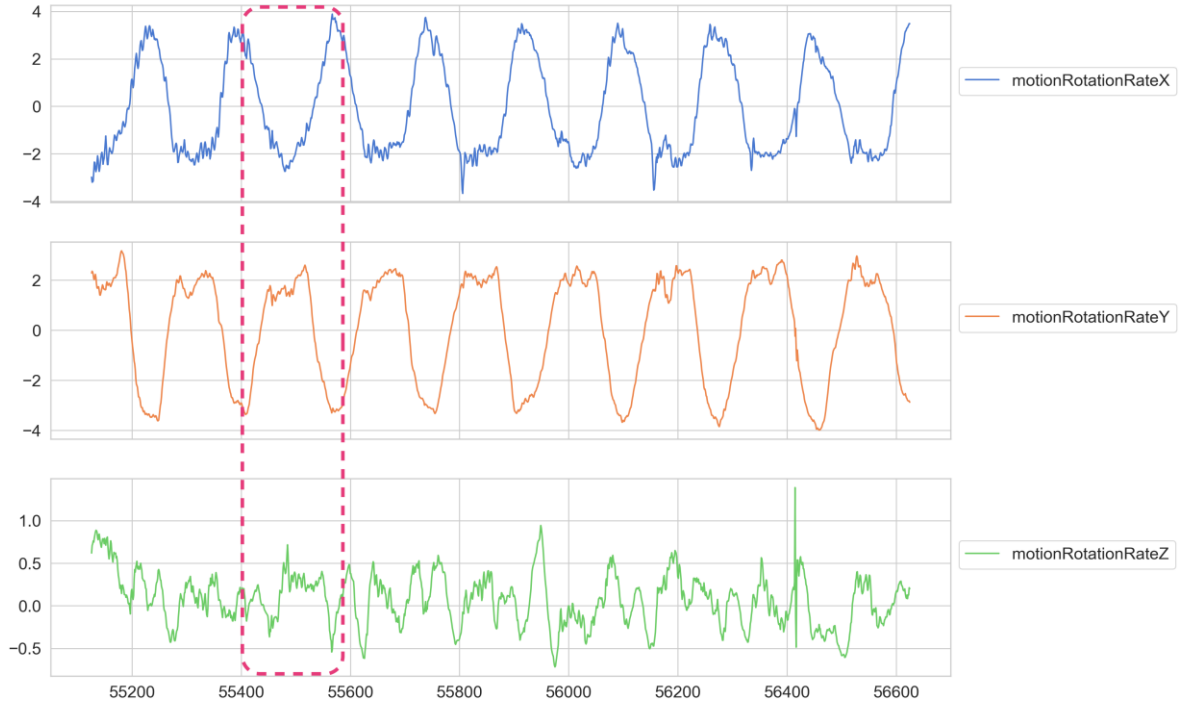

Рисунок 4.1 – Довжина середнього сегменту одного повторення ФА

Відповідно до загальних практик використаних при інших суміжних дослідах з використанням показників акселерометра чи гіроскопа, можна визначити загальні тенденції розподілу опцій відносно інших гіперпараметрів системи. Для виконання задач пошуку оптимального значення візьмемо за відправну точку комбінації значень параметрів зазначені у таблиці 4.1.

Таблиця 4.1 – початкові комбінації значень гіперпараметрів

Гіперпараметр	Перша опція	Друга опція	Третя опція
Dropout rate	0,2	0,35	0,5
Learning rate	0,001	0,0005	0,0001
Batch size	64	96	128

Наведемо більш детальний опис кожного з параметрів.

Dropout (або рівень забування) – це техніка, яка використовується під час навчання нейромереж для запобігання перенавчання. Під час тренування певний відсоток нейронів у шарі випадковим чином вимикаються. Це означає, що залишені нейрони змушені вивчити важливі особливості вхідних даних, а не залежати від наявності всіх нейронів для прогнозування. У багатьох дослідах з використанням засобів глибокого навчання зазначається, що використання даного механізму покращує загальну здатність нейромереж до узагальнення та запобігає перенавчанню, дозволяючи мережі ефективно працювати з новими та невідомими даними без втрати точності.

Крок навчання (learning rate) – гіперпараметр, який визначає, наскільки сильно ваги нейронної мережі модифікуються відносно градієнту втрат під час зворотного поширення помилки. Встановлення занадто високого кроку навчання може спричинити неможливість мережі досягнути мінімуму, тоді як низька швидкість навчання може призвести до повільної збіжності або сприйняття локального мінімуму за глобальний.

Важливо визначити оптимальне значення кроку крок навчання при навчанні нейронної мережі, щоб забезпечити швидку та точну збіжність мережі. Оптимальний крок навчання може залежати від різних чинників, включаючи розмір мережі, складність задачі, яку необхідно вирішити, і кількість та якість тренувальних даних.

Існують різні техніки, такі як чергування кроків навчання та адаптивні кроки навчання, які можуть використовуватися для налаштування мережі під час тренування. Ці техніки можуть допомогти покращити продуктивність мережі та запобігти застряганню в локальних мінімумах.

Під час процесу навчання нейронної мережі дані обробляються пакетами (batch), а не всіма одразу. Вибір розміру пакету може суттєво вплинути на процес навчання.

Більші розміри пакетів можуть призвести до швидшої збіжності, оскільки ваги оновлюються рідше. Однак це також може призвести до отримання підоптимальних рішень через нестачу точності у тонкій настройці ваг. Менші розміри пакетів, натомість, можуть призвести до повільнішої збіжності, але в результаті надають більш точних рішень.

Важливо враховувати специфіку конкретної проблеми та доступне обладнання при виборі відповідного розміру пакету. Відповідно до наукової літератури гарною практикою є експериментування з різними розмірами пакетів та оцінювання їх продуктивності для визначення оптимального значення.

4.2 Оптимізація гіперпараметрів системи

Спочатку проведемо порівняльний аналіз моделі з різними правилами сегментації вхідних даних. У свою чергу для тестування перехресної сегментації у якості мінімального кроку зміни будемо вважати 10% від мінімальної довжини сегменту. Таким чином рівень перекриття буде 90%, при якому шанси на випадкову втрату важливих граничних патернів буде

зведено до нуля. Результати пошуку оптимальних правил сегментації наведено на рисунку 4.2. Порівняльний аналіз правил сегментації був проведений з використанням двох шарової GRU мережі.

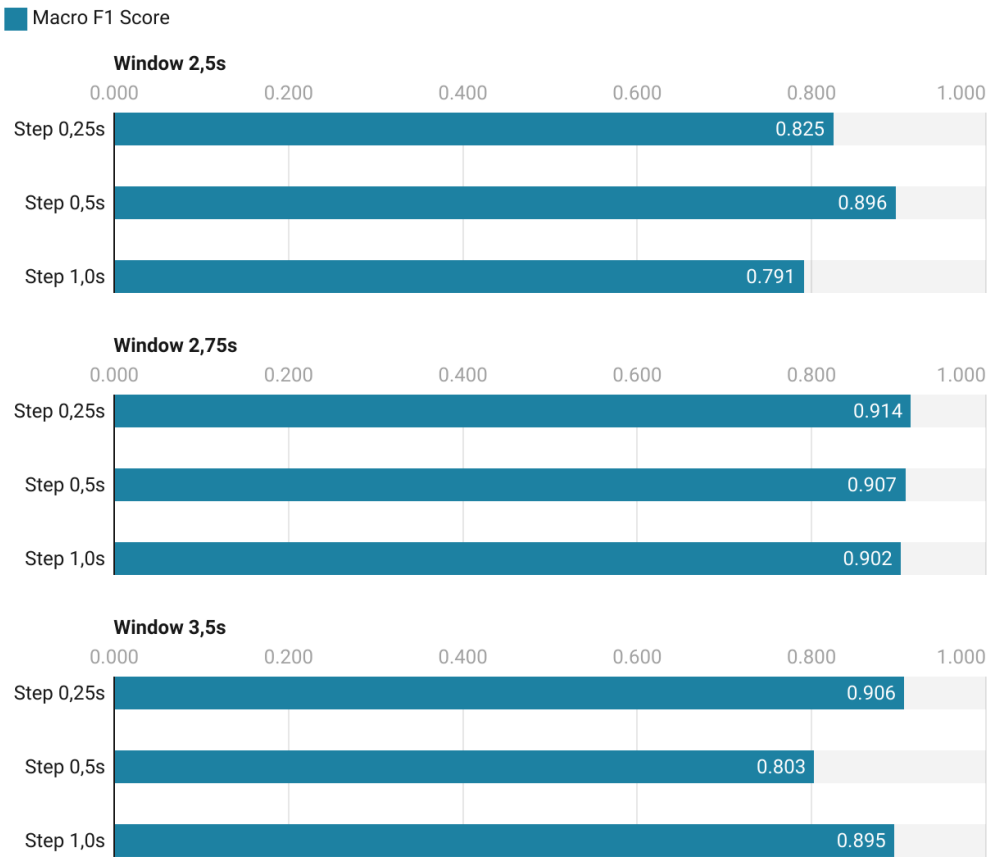

Рисунок 4.2 –Метрики при порівнянні правил сегментації

Як можна спостерігати на графіку, результати є не сильно диференційними, але все одно є змога виділити загальну тенденцію, що інтервал у 2.75 секунди з кроком у 0.25 секунди дає найбільшу ефективність класифікатора з найбільшою стійкістю до рівня перехресності. Таким чином встановлюючи довжину кожного сегменту на 10% довшою за період повторення найдовшої вправи є оптимальний варіантом визначення довжини сегменту.

Крок сегментації в оптимальному варіанті же становить 9%, тобто сегментація сирих даних відбувається з перекриттям на рівні 91%. В теорії

перекриття на рівні 50% було б достатньо для мінімізації ризиків втрати граничних патернів, але на практиці маємо найкращі результати при 91% перекриття. Можливо це зумовлено недостатньою кількістю вхідних даних, і при меншому рівні перекриття система не отримує достатню кількість сегментів для достатнього навчання. У випадку наявності більшої кількості вхідних даних можливо буде раціональним зменшити рівень перекриття сегментів до 50%. Наразі, через наявність обмежених даних лише однієї особи, рівень перекриття в 91% не тільки зведе вірогідність втрати граничних патернів до нуля, але і також збільшить кількість навчальних даних.

Для аналізу ефективності системи також було проведено тестування з різною кількістю нейронів на кожному з шарів мережі (наведено на рисунку 4.3). Усі досліди показали високу точність класифікації, але враховуючи гарантоване розширення переліку типів ФА у майбутньому слід зупинитися на варіанті з 128 нейронів на кожному шарі. Таким чином ми досягаємо оптимальної точності на наявних вхідних даних, і маємо певний запас потенціалу у випадку розширення кількості класів. Адже навіть при класифікації дев’яти типів ФА ми маємо певні просадки у точності при 64 нейронах. У випадку ж 256 нейронів, у даному випадку вони виконують поставлену задачу з такою ж точністю, але при цьому вимагають у рази більше обчислень. Для виконання проектування та тестування є можливим підвищення кількості нейронів на кожному з шарів системи, але у даному випадку це призведе до зайвого ускладнення системи. По-перше додавання надлишкових нейронів може слугувати причиною до перетренованості моделі, що призведе до зменшення загальної точності мережі на тестових даних, а також до збільшення необхідних розрахунків під час тренування та виконання саме класифікації. А через особливість ПО, де всі обчислення потенційно будуть виконуватися на портативних особистих пристроях користувачів, треба забезпечити максимальну ефективність системи при мінімальних витратах апаратних ресурсів. Таким чином, при виконанні

наступних дослідів, кількість нейронів на кожному шарі було встановлено на рівні 128 нейронів.

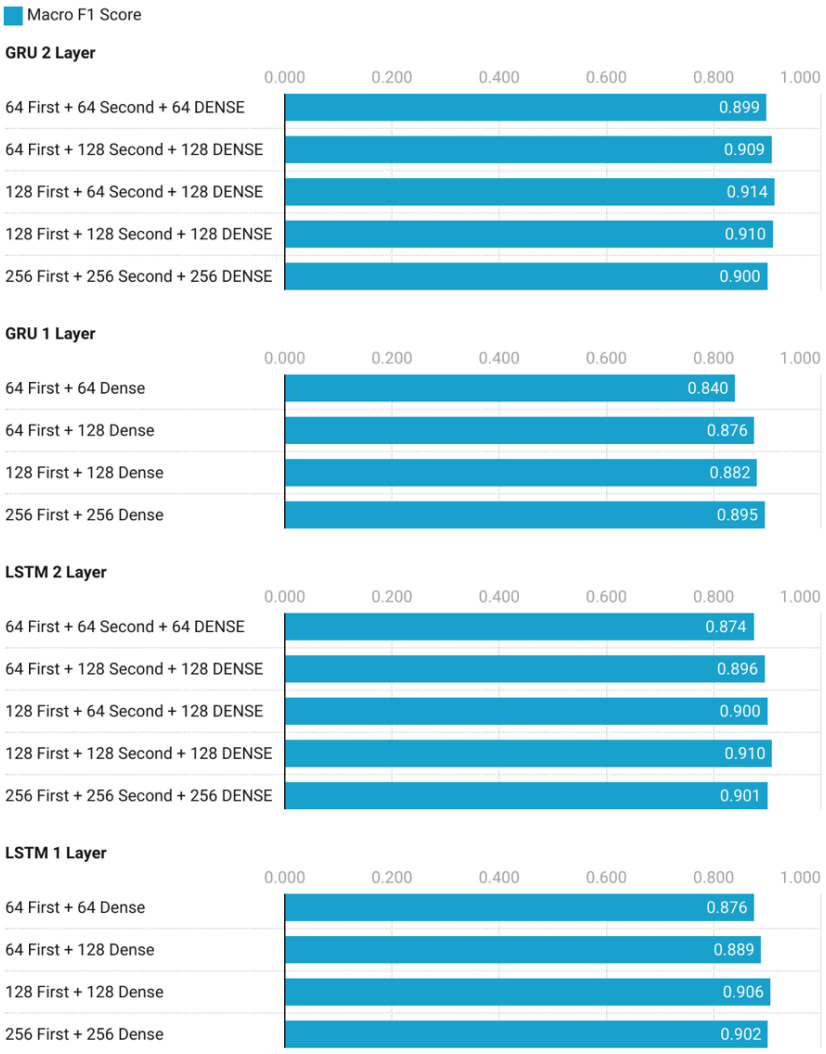

Рисунок 4.3 – Результати тестування різної кількості нейронів на кожному шарі

Також слід зазначити особливість архітектури у вигляді окремого Dense шару, який представляє з себе шар з N нейронів які представляють з себе елементарні логічні блоки з ReLu функцією активації, яка отримуючи на вхід від'ємне число повертає значення 0, у інших же випадках повертає аргумент функції. Але логіка роботи кожного з ReLu нейронів також підлягає оптимізації під час проведення навчання, то ж у результаті кожен

нейрон Dense шару буде мати власну логіку роботи і граничні значення спрацювання.

Враховуючи усі можливі комбінації глобальних параметрів, у якості порівняльних були виділені граничні показники параметрів (див. таблицю 4.1). Таким чином на рисунку 4.4 наведено результат Macro F1 Score для наборів параметрів (комбінації перших, других та третіх альтернатив). Через велику кількість суміжних комбінацій значень цих параметрів, результати кожного досліду наводити немає сенсу, а оптимальні показники параметрів наведені у таблиці 4.2.

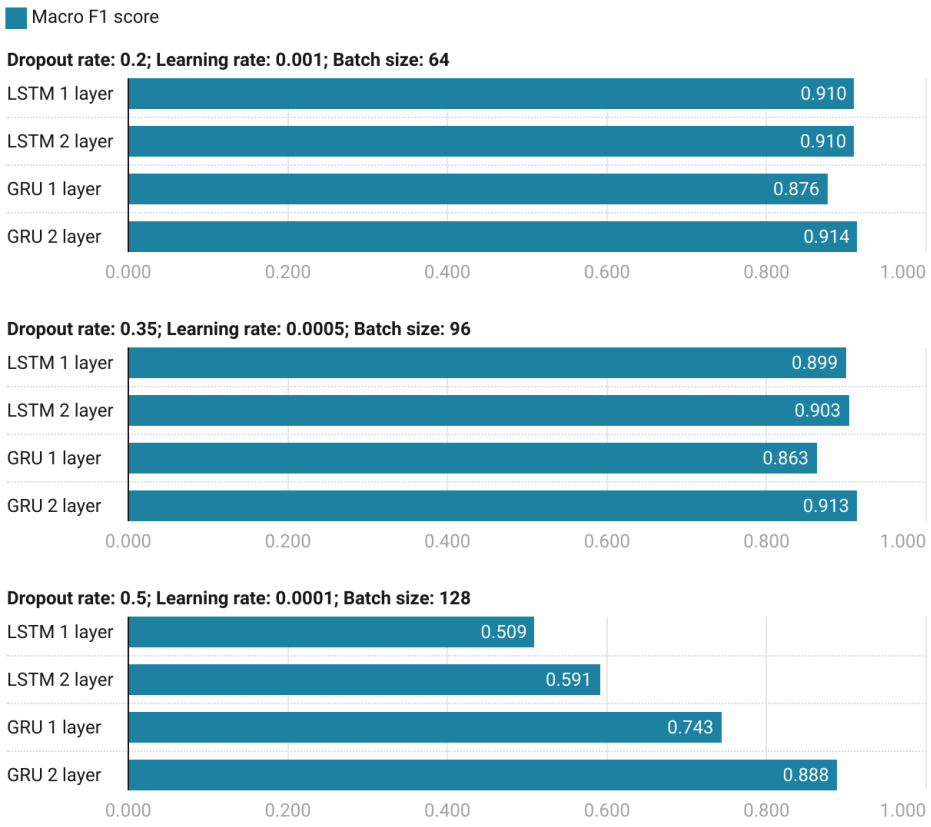

Рисунок 4.4 – Графік залежності точності системи від значень гіперпаметрів

Аналізуючи ж наведений графік точності системи при комбінації різних архітектур та значень глобальних параметрів, можна визначити, що значення навчального рівня мережі достатньо сильно впливає на загальну

точність системи, і у даному випадку чим менше значення показника, тим менш точною стає система. Але ця залежність не є лінійною, тому збільшувати показник більше 0.001 не має сенсу, адже ми можемо отримати проблему зникаючого градієнту під час оптимізації через занадто великий крок оптимізації. У даному випадку при мінімальному значенні 0.0001 ми, скоріш за все, отримали ситуацію коли мережа під час навчання натрапила на локальний мінімум функції оптимізації, і не змогла вийти з нього через малий крок оптимізації, що призвело до значущого зменшення точності мережі на тестових даних.

У якості кращих комбінацій для кожного типу архітектури можна виділити комбінації які зазначені у таблиці 4.2.

Таблиця 4.2 – Оптимальні комбінації значень гіперпараметрів для різних архітектур

Network type	Learning rate	Batch size	Dropout rate	Macro F1 score	Specificity	Sensitivity
GRU 1 Layer	0.001	64	0.35	0.8760	0.8708	0.8680
GRU 2 Layer	0.0005	96	0.5	0.9298	0.9192	0.9204
LSTM 1 Layer	0.0005	64	0.35	0.9103	0.9083	0.9072
LSTM 2 Layer	0.001	64	0.2	0.9193	0.9893	0.9180

Таким чином, найефективнішою мережею є мережа з двома GRU шарами з швидкістю навчання 0,0005, розміром пакету 96 та dropout-рівнем 0,5, яка досягає макро F1-оцінки 0,9298, специфічності 0,9192 та чутливості 0,9204.

4.3 Аналіз ефективності системи в залежності від частоти дискретизації вхідних даних

Розглядаючи задачу мінімізації апаратних витрат у процесі функціонування системи визначення ФА людини, слід також враховувати

можливість зменшення необхідної частоти дискретизації показників сенсорів. У випадку якщо ефективність мережі буде на тому ж самому рівні при використанні показників з частотою дискретизації 100 Гц та 25 Гц, не має сенсу продовжувати використання даних при частоті 100 Гц, адже це змушує виконувати у 4 рази більше обчислень для отримання того ж результату який можна отримати і при 25 Гц.

Виконавши досліди з аналізу точності системи відповідно до використаних архітектур та частот дискретизації сигналу ми отримали наступні результати (рисунок 4.5).

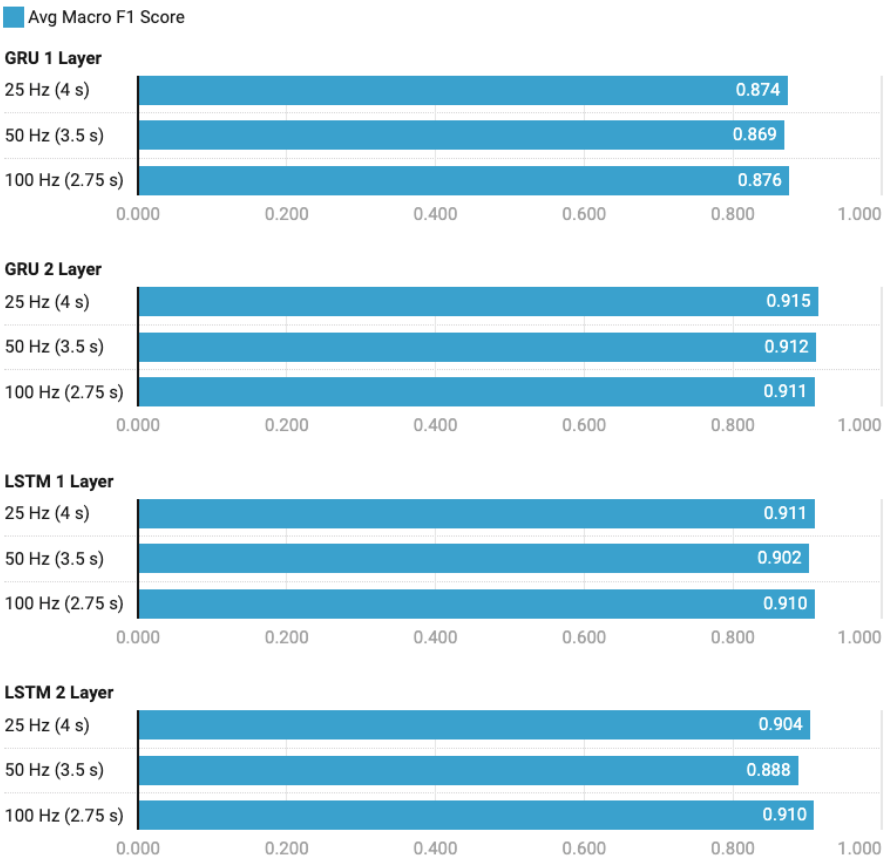

Рисунок 4.5 – Залежність точності системи від частоти дискретизації вхідних даних за різних архітектур

В результаті проведення дослідів по зменшенню частоти дискретизації показників сенсорів, було виявлено певну залежність між

частотою дискретизації та довжиною сегменту даних, яка впливає на точність результуючої моделі. У минулих дослідах правил сегментації сирих даних були використані дані з частотою 100 Гц. Попередньо можна сказати що найбільш оптимальною довжиною сегменту є 2.75 секунди при частоті дискретизації 100 Гц. Але при проведенні дослідів було помічено зменшення загальної точності мережі при зменшенні частоти дискретизації показників і збереженні довжини сегменту. Було вирішено збільшити довжину сегменту зі збереженням рівню перекривання більше 50%, таким чином збільшивши довжину сегменту з 2.75 секунд до 4 секунд. Таким чином при аналізі ефективності системи вона була більш стабільною і надавала в середньому вищі показники точності. Також слід врахувати необхідність збереження рівня перехресної сегментації на високому рівні. У проведених дослідах рівень перехресної сегментації було встановлено на 75%, що дало оптимальний результат. Як вже зазначалося, з більшою кількістю даних рівень перехресної сегментації теоретично можна зменшити до рівня 50%, але на даному етапі, з використання даних однією особи, рівень перехресної сегментації треба тримати на більш високому рівні для компенсації недостатньої кількості даних.

Таким чином при виконанні процесу збільшення довжини кожного сегменту при зменшенні частоти дискретизації сигналу можна досягти мінімальної втрати точності системи незалежно від частоти дискретизації системи. Але, як зазначалося раніше, використовувати сигналу частотою менше 25 Гц не рекомендується, адже при частоті нижче 25 Гц [6] йде нелінійне зменшення точності систем які використовували дані акселерометра та гіроскопа. Та і відповідно до технічного завдання, використання сигналу з частотою дискретизації 25 Гц є значним покращенням відносно використання 100 Гц сигналу. Отже загальні апаратні вимоги до мінімально можливої частоти оновлення показників можна зменшувати до 25 Гц.

Але слід врахувати потенційну можливість системи до втрати точності при подальшому збільшенні кількості типів ФА які вона може класифікувати. Адже на даному етапі досліджується невелика кількість достатньо ізольованих один від одного тренувань, при збільшенні класів системи можуть бути кореляції між схожими один на одного типів ФА.

4.4 Оптимальний варіант програмної реалізації класифікатора

Маючи результати всіх досліджень тепер є змога визначити оптимальний варіант програмної реалізації проектованої системи. По-перше треба визначити вимоги до вхідних даних системи.

В результаті проведеного дослідження було проведено порівняльний аналіз роботи класифікатора з використанням трьох частот дискретизації вхідного сигналу. Як підсумок даного досліду можна вважати, що підвищення частоти дискретизації ніяким чином, або мінімально, впливає на точність роботи системи. Але при цьому треба враховувати необхідність підвищення довжини кожного сегменту при зменшенні частоти, та збору більшої кількості даних для навчання системи. Адже чим рідше показники сенсорів будуть записуватися, тим більше реального часу треба виконувати окремі вправи для забезпечення системи повним датасетом. Тож оптимальним значенням частоти дискретизації сигналу є 25 Гц.

Щодо дослідів глобальних гіперпараметрів системи маємо наступні результати. При дослідженні використання різного кроку навчання (learning rate) виявилася невелика різниця між кроком 0.001 та 0.0005, але найменший варіант, 0.0001, показав набагато гірший результат за своїх конкурентів. Таким чином при використанні мінімального значення у якості кроку навчання призвело до сприйняття локального мінімуму функції втрат як глобального, через що глобальна оцінка точності системи показує критично малі показники. Таким чином обираючи між двома першими опціями кроку навчання (див. таблицю 4.1) можна зупинитися на першому значенні

(0.001). Але враховуючи результати дослідження наведені у таблиці 4.2, деякі моделі мають кращу точність при навчанні з першою опцією, другі моделі мають кращі результати з другою опцією. Припускаю, що отримана різниця у досліді була у рамках похибки, і при проведенні більшої кількості дослідів середнє значення точності при двох значеннях було б більш рівним. Таким чином, для оптимізації процесу навчання пропонується використання динамічного кроку навчання, де на перших поколіннях навчання буде використовуватися більше значення, і при зменшенні значення функції втрат значення кроку буде також змінюватися до 0.0005. Це дозволить потенційно скоротити час навчання системи без втрати точності результуючої моделі.

Аналізуючи можливі опції рівню забування (dropout rate) можна визначити що наразі різні показники не мають великого впливу на ефективність моделі. Це обумовлено невеликою кількістю даних та, що ймовірніше, природою наявних даних. Оскільки даний датасет був зібраний одною людиною, та включає патерни рухів тільки однією людини, то цей показник не є критичним, адже усі рухи які виконує одна людина під час однієї ФА є максимально схожими один між одним. Але у випадку розгляду системи яка навчалася на даних які збиралися декількома особами, цей показник набув би великого впливу на результат. Адже дуже малоймовірно що окремі люди будуть виконувати окремі вправи однаково. Звісно загальний патерн рухів буде схожий, але буде різний період повторення, екстремуми показників різних координат та сама інтенсивність. У такому випадку треба врахувати значення dropout rate відносно характеру використання проектованої системи .У випадку використання системи великою кількістю людей, збільшення значення гіперпараметра є оптимальним варіантом (значення ~0.5). Якщо ж система орієнтована на обмежену аудиторію яка приймала участь при навчанні системи, то є сенс залишити значення параметру на невисокому рівні (від 0.2 до 0.35).

Аналізуючи отримані результати дослідження різних опцій гіперпараметру розміру пакету при оптимізації мережі було виявлено певну залежність. Річ у тім що чи більший розмір пакету тим гірший результат ми отримуємо в результаті навчання. Незважаючи на чималу кількість поколінь навчання системи, результат при використанні великого пакету не є задовільним. Це скоріш за все обумовлено відсутністю можливості тонкого налаштування ваг нейронів, через занадто великі обсяги одного кроку оптимізації, таким чином система не має здатності визначити оптимальні ваги, оскільки вони змінюються все більшим набором за ітерацію, і потрапляння на вдалу комбінацію становиться настільки випадковим явищем, що в рамках проектування його можна вважати неможливим. Таким чином значення розміру пакету можна встановлювати у мінімальне значення для усіх типів архітектур, при умові проведення проектування та навчання системи на окремому апаратному засобі який матиме достатню обчислювальну здатність. У випадку розгляду сценарію динамічного донавчання системи на особистих пристроях користувачів, або з метою мінімізації необхідних обчислень під час навчання, рекомендується встановлювати значення batch size не більше 96, адже при значенні більше йде значне погіршення якості результуючої моделі.

Враховуючи визначені оптимальні значення та комбінації гіперпараметрів системи тепер ми можемо визначити оптимальну архітектуру до розробки проектованої мережі. Отримані результати аналізу не є такими однозначними, адже розбіжність між різними архітектурами при використанні оптимальних комбінацій значень гіперпараметрів дають дуже схожі результати, тож рішення щодо кращої альтернативи приймається на основі суміжних висновків. Під час проведення дослідження різних архітектур, під час виведення середнього значення ефективності аналізуючи декілька ітерацій навчання, було помічено значно більшу стабільність результатів моделей з використанням GRU блоків. Враховуючи також символічну перевагу у точності системи з GRU блоками

над LSTM блоками на використаній кількості даних, можна припустити схожу тенденцію при використанні набагато більшої кількості даних, адже чим більше даних та класів, тим більше семантичне навантаження на мережу, що неодмінно призводить до погіршення результатів. Тож у даному випадку використання GRU блоків у рекурентній мережі є більш перспективним методом розробки. Також слід врахувати момент оптимізації апаратних витрат під час навчання та функціонування системи. Відповідно до концепції GRU блоків, вони мають на один функціональний елемент менше аніж LSTM блок (відсутній вихідні ворота), що випливає у зменшенні кількості коефіцієнтів мережі та кількості необхідних розрахунків на кожній ітерації роботи системи. Тож це рішення є одночасно і більш привабливим з точки зору точності та енергоефективності результуючої моделі.

Відповідно до кількості шарів, тут все достатньо однозначно. Хоча різниця між одношаровою та двошаровою LSTM мережею є мінімальною, у випадку GRU одношарова альтернатива має гірші результати точності, хоча вона і використовує значно менше апаратних ресурсів. Тим не менш, враховуючи природу вхідних даних, а саме потенційну складність даних у випадку розширення кількості типів ФА людини, де можливі дуже схожі патерни рухів, результуюча модель повинна мати певний запас ресурсів на випадок підвищення складності задачі, що без сумніву буде мати місце. Таким чином, розглядаючи вибір оптимальної архітектури мережі для реалізації нейромережевого класифікатору визначення типу ФА людини для автоматизації бізнес-процесів фітнес-клубу варто використовувати рекурентну нейронну мережу з двома шарами GRU блоків, з використанням довжини сегменту 2.75 секунди (при частоті дискретизації вхідних даних 100 Гц), розміром пакету 64 або 96, рівнем забуванням 0.5, та кроком навчання 0.0005 (або з використанням динамічного кроку з початковим більшим значенням з його поступовим зменшенням).

ВИСНОВКИ

В результаті виконання науково-дослідної роботи було проведено аналіз, проектування та розробка прототипу інформаційно-аналітичної системи фітнес-клубу для визначення типу ФА людини. Було проведено аналіз предметної галузі стосовно бізнес функцій клубу та визначені основні вимоги до проектованої системи. Було також проведено аналіз існуючих конкуруючих програмних додатків зі схожими задачами, для виявлення кращих практик та недоліків конкурентів.

Окрім того було проведено порівняльний аналіз засобів реалізації прототипу системи та проведено аналітики щодо можливих варіантів реалізації, з наведенням кращих практик які варто використовувати при вирішенні задач подібної природи. Для проведення цього аналізу було розроблено окремий додаток для збору показників просторових сенсорів з особистих пристроїв, таких як гіроскоп та акселерометр.

Проведено аналіз двох методів реалізації нейромережевого класифікатора з використанням LSTM та RNN блоків, та наведено порівняльну характеристику цих двох методів. Визначені основні апаратні вимоги до окремих компонентів проектованої системи зумовлені вимогами до показників сенсорів, використаних класифікатором.

Для забезпечення якомога раціонального порівняльного аналізу відібраних методів реалізації системи були визначені основні порівняльні метрики за якими буде проводитися визначення кращої альтернативи.

В результаті порівняльного аналізу було досягнуто зменшення необхідної частоти дискретизації сигналу у 4 рази, та зменшення кількості необхідних розрахунків під час навчання системи та виконання класифікації типу ФА.

Розглянувши отримані результати, можна стверджувати що при індивідуальному підході до кожного користувача та оптимальної кількості типів ФА можливо досягти достатньо високої точності класифікації ФА

людини, що виконує поставлені цілі до даної ІАС. Варіант же використання даної системи для більшої кількості користувачів є предметом окремого дослідження, але враховуючи чималий потенціал використаних у даній роботі технологій, є певна впевненість у позитивних результатах майбутніх досліджень.

Для подальшого удосконалення системи варто провести додатковий порівняльний аналіз динамічного кроку навчання системи, для визначення раціональності його використання у фінальному продукті. Проведення аналізу ефективності системи з використанням більшої кількості типів ФА за участі більшої кількості людей для аналізу стабільності системи при збільшенні навчальних даних та класів.

Відповідно дана інформаційно-аналітична система є невід`ємною частиною більшої ІАС фітнес-клубу, яка включає до себе більший перелік автоматизованих бізнес-функцій, то ж фінальним етапом розробки цієї системи буде інтеграція цієї системи у якості окремого модуля до загального додатку фітнес-клубу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bonomi A. G., Goris A. H. C., Yin, B., Westerterp K. R. Detection of type, duration, and intensity of physical activity using an accelerometer. *Medicine & Science in Sports & Exercise*. 2009. Vol. 43. P. 1770–1777. DOI: 10.1249/MSS.0b00b0b13e33e3e181a11a2a4536

2. Gyllensten I. C., Bonomi A. G. Identifying types of physical activity with a single accelerometer: evaluating laboratory-trained algorithms in daily life. *IEEE Trans. Biomed. Eng*. 2011. Vol 58. P.2656–2663. DOI: 10.1109/TBME.2011.2160723

3. Hoda Allahbakhshi, Timo Hinrichs, Haosheng Huang, Robert Weibel. The Key Factors in Physical Activity Type Detection Using Real-Life Data: A Systematic Review. *Computational Physiology and Medicine*. 2019. Vol. 10. DOI: 10.3389/fphys.2019.00075

4. Adaskevicius R. Method for recognition of the physical activity of human being using a wearable accelerometer. *Elektron ir Elektrotech*. 2014. Vol 20. P. 127–131. DOI: 10.5755/j0j1.eee.20.5.7113

5. Preece S. J., Goulermas J. Y., Kenney L. P. J., Howard D., Meijer K., Crompton R. Activity identification using body-mounted sensors – a review of classification techniques. *Physiol*. 2009. Vol. 30. DOI: 10.1088/09677-3334/30/4/R0R1

6. De Vries S. I., Garre F. G., Engbers L. H., Hildebrandt V. H., Van Buuren S. Evaluation of neural networks to identify types of activity using accelerometers. *Med. Sci. Sport Exerc*. 2011. Vol. 43. P. 101–107. DOI: 10.1249/MSS.0b00b0b13e33e3e181e11e5e797d77d

7. Barshan B., Yuksek M. C. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *Comput. J*. 2014. Vol. 57. P. 1649–1667. DOI: 10.1093/comjnl/bxt0t75

8. Kwak K., Lee M. “Physical activity classification using tsk-type neuro- fuzzy classifier with GK clustering” in Proceedings on the International

Conference on Artificial Intelligence (ICAI) The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing. *WorldComp*. 2012. Las Vegas, NV.

9. Garcia-Ceja E., Brena R. F. Activity recognition using community data to complement small amounts of labeled instances. *Sensors.* 2016. DOI: 10.3390/s16060877

10. Gao C., Neil D., Ceolini E., Liu S.C., Delbruck T. Delta. RNN: A power-efficient recurrent neural network accelerator. In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA. 2018. P. 21–30.

11. Yu S. Residual Learning and LSTM Networks for Wearable Human Activity Recognition Problem. In Proceedings of the 2018. 37th IEEE Chinese Control Conference (CCC). Wuhan, China. 2018. P. 9440–9447.

12. De Vries S. I., Garre F. G., Engbers L. H., Hildebrandt V. H., Van Buuren S. Evaluation of neural networks to identify types of activity using accelerometers. 2011. *Med. Sci. Sport Exerc*. Vol. 43. P. 101–107. DOI: 10.1249/mss.0b013e3181e5797d

13. Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertainty Fuzziness Knowl.-Based Syst.* 1998. P. 107–116. DOI: 10.1142/S0218488598000094

14. Bengio Y., Simard P., Frasconi P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw*. 1994. Vol 6. P. 157–166. DOI: 10.1109/72.279181

15. Luna-Perejón F., Civit-Masot J., Muñoz-Saavedr L., Durán-López L., Amaya-Rodríguez I., Domínguez-Morales J.P., Vicente-Díaz S., Linares-Barranco A., Civit-Balcells A., Domínguez-Morales M. Sampling Frequency Evaluation on recurrent neural networks Architectures for IoT Real-time Fall Detection Devices. International Joint Conference on Computational Intelligence (INSTICC). Vienna, Austria. 2019. P. 536–541.

16. Sokolova M., Lapalme G. A systematic analysis of performance

measures for classification tasks. *Inf. Process. Manag*. 2009. Vol. 45. P. 427–437. DOI: 10.1016/j.ipm.2009.03.002

ДОДАТОК А

Графічні матеріали

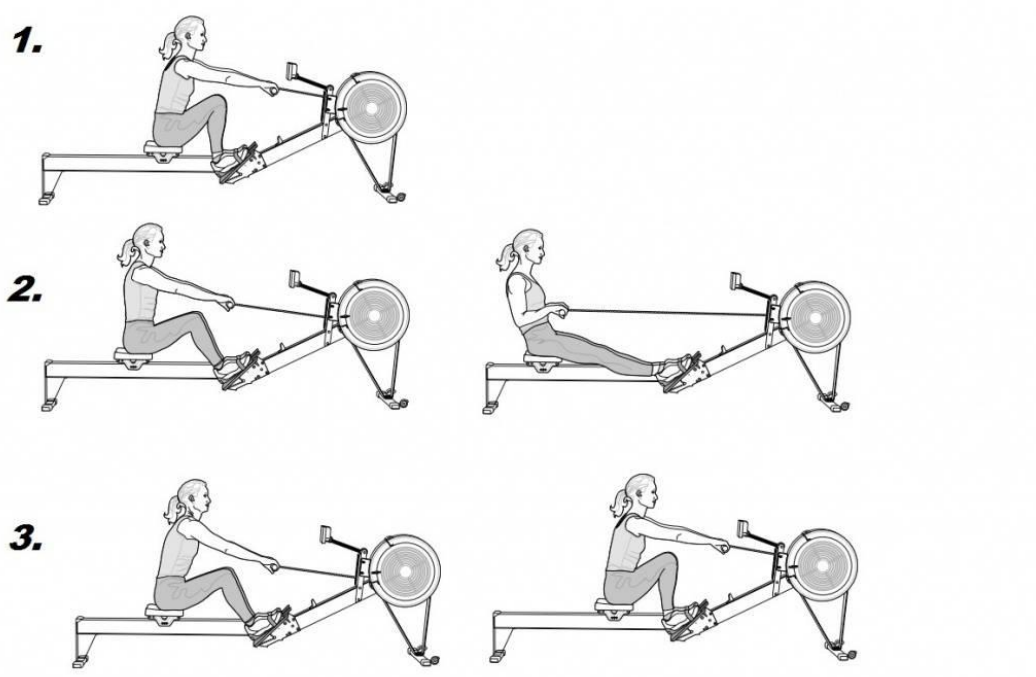

Рисунок А.1 – ілюстрація використання тренажеру греблі

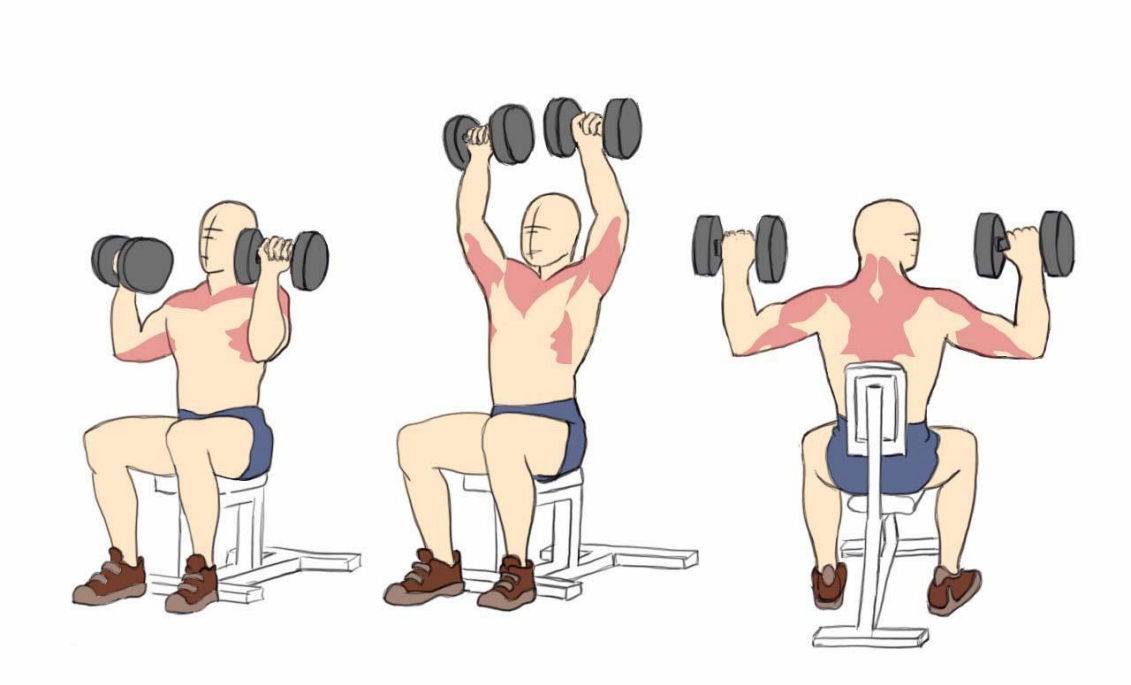

Рисунок А.2 – ілюстрація виконання жиму догори гантелей сидячи

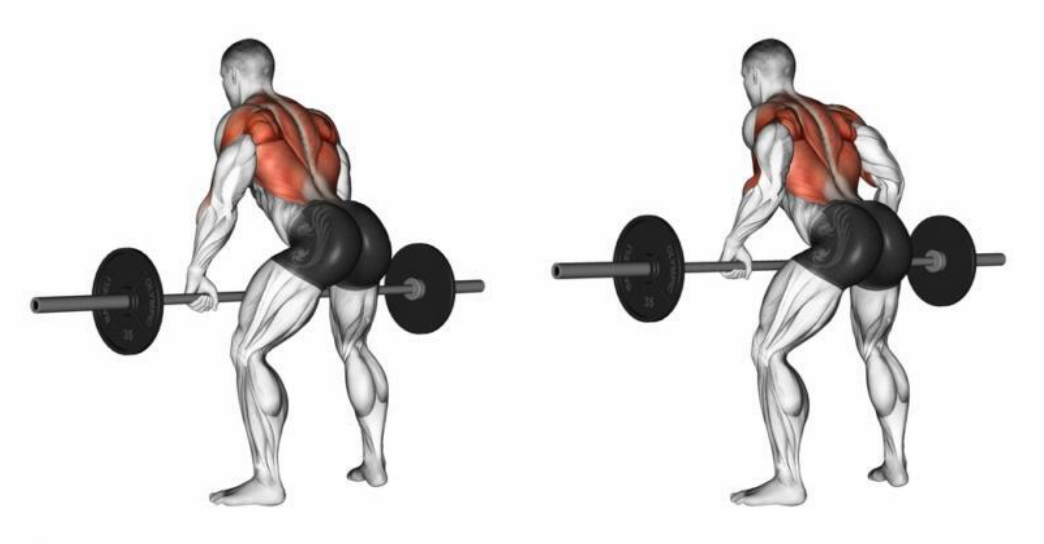

Рисунок А.3 – ілюстрація виконання тяги штанги до поясу стоячи

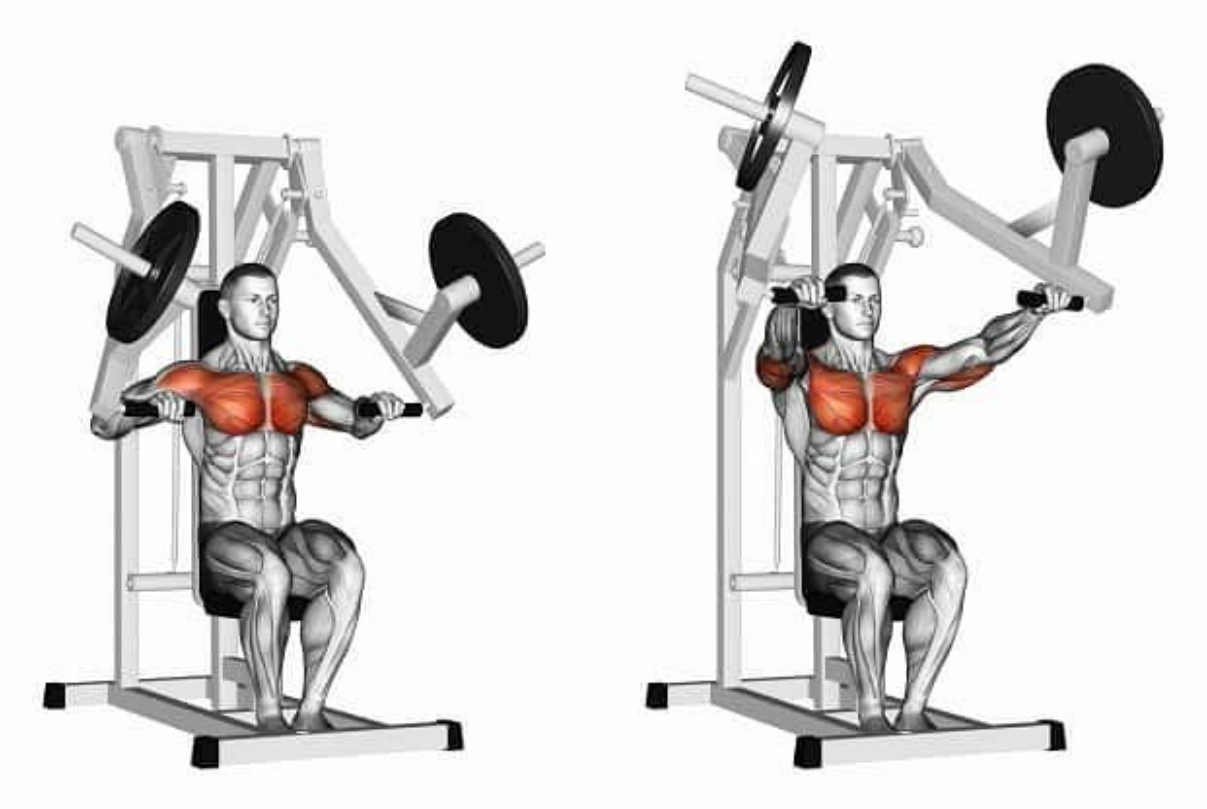

Рисунок А.4 – ілюстрація використання тренажеру для жиму на груди

Рисунок А.5 – ілюстрація виконання тяги тросу к поясу сидячи

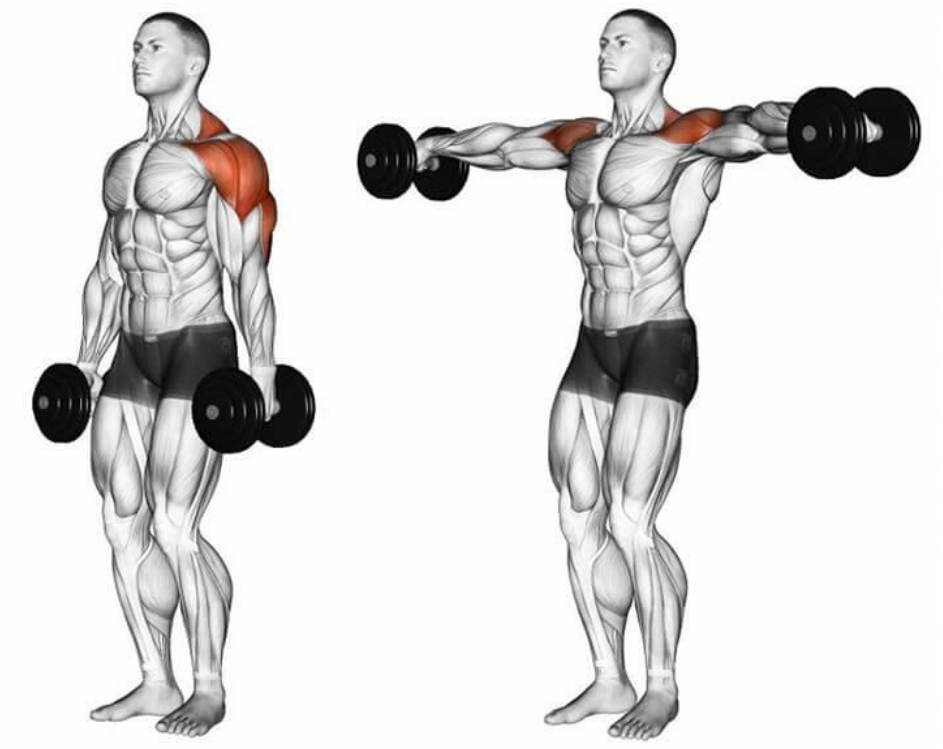

Рисунок А.6 – ілюстрація виконання розведення гантелей у бік

ДОДАТОК Б

Текст програми

Python код програмної реалізації класифікатора з LSTM блоками

```
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from pandas.plotting import register_matplotlib_converters

%matplotlib inline
%config InlineBackend.figure_format='retina'

register_matplotlib_converters()
sns.set(style='whitegrid', palette='muted', font_scale=1.5)
rcParams['figure.figsize'] = 22, 10

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
tf.random.set_seed(RANDOM_SEED)

column_names = ['loggingTime', 'deviceID', 'batteryLevel', 'label',
'motionTimestamp_sinceReboot', 'motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ', 'motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ', 'groupCode']
df = pd.read_csv('grouped_united_data_v6.csv', header=0, names=column_names,
sep=';')
df.dropna(axis=0, how='any', inplace=True)

normalizedGroupedDf = df.sort_values(['groupCode','loggingTime',
'motionTimestamp_sinceReboot'], ascending=True).groupby('groupCode',
as_index=False, group_keys=False).apply(lambda x:
x.iloc[int(x.groupCode.size*0.1):int(x.groupCode.size*0.9)]).reset_index()
trainDf = normalizedGroupedDf.groupby('label', as_index=False,
group_keys=False).apply(lambda x: x.iloc[int(x.label.size *
```

```
0.2):int(x.label.size)]).reset_index()
testDf = normalizedGroupedDf.groupby('label', as_index=False,
group_keys=False).apply(lambda x: x.iloc[:int(x.label.size *
0.2)]).reset_index()

def plot_activity_acc(label, df):
    data = df[df['label'] == label][['motionUserAccelerationX',
'motionUserAccelerationY', 'motionUserAccelerationZ']][
           :1500]
    axis = data.plot(subplots=True, figsize=(16, 12),
                     title=label)
    for ax in axis:
        ax.legend(loc='lower left', bbox_to_anchor=(1.0, 0.5))


def plot_activity_rot(label, df):
    data = df[df['label'] == label][['motionRotationRateX',
'motionRotationRateY', 'motionRotationRateZ']][:1500]
    axis = data.plot(subplots=True, figsize=(16, 12),
                     title=label)
    for ax in axis:
        ax.legend(loc='lower left', bbox_to_anchor=(1.0, 0.5))

from sklearn.preprocessing import RobustScaler
scale_columns = ['motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ','motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ']
scaler = RobustScaler()
scaler = scaler.fit(trainDf[scale_columns].values)
trainDf.loc[:, scale_columns] =
scaler.transform(trainDf[scale_columns].to_numpy())
testDf.loc[:, scale_columns] =
scaler.transform(testDf[scale_columns].to_numpy())

from scipy import stats

def create_dataset(X, y, time_steps=1, step=1):
    Xs, ys = [], []
    for i in range(0, len(X) - time_steps, step):
        v = X.iloc[i:(i + time_steps)].values
        labels = y.iloc[i: i + time_steps]
```

```
        Xs.append(v)
        ys.append(stats.mode(labels)[0][0])
    return np.array(Xs), np.array(ys).reshape(-1, 1)

TIME_STEPS = 275
STEP = 25

X_train, y_train = create_dataset(
    trainDf[['motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ','motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ']],
    trainDf.label,
    TIME_STEPS,
    STEP
)

X_test, y_test = create_dataset(
    testDf[['motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ','motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ']],
    testDf.label,
    TIME_STEPS,
    STEP
)

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore', sparse=False)
enc = enc.fit(y_train)
y_train = enc.transform(y_train)
y_test = enc.transform(y_test)
from tensorflow import keras
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
```

```
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))

def specificity(y_true, y_pred):
    neg_y_true = 1 - y_true
    neg_y_pred = 1 - y_pred
    fp = K.sum(neg_y_true * y_pred)
    tn = K.sum(neg_y_true * neg_y_pred)
    return tn / (tn + fp + K.epsilon())

checkPointPath =
'/Users/skrekoza/PycharmProjects/tensorflow_proj/lstm_2_layer_best_test/'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkPointPath,
    save_weights_only=True,
    monitor='f1_m',
    mode='max',
    save_best_only=True)

model = keras.Sequential()
model.add(
    keras.layers.Bidirectional(
      keras.layers.LSTM(
          units=128,
          input_shape=[X_train.shape[1], X_train.shape[2]],
          return_sequences=True
      )
    )
)
model.add(keras.layers.Dropout(rate=0.2))
model.add(
    keras.layers.Bidirectional(
      keras.layers.LSTM(
          units=64
      )
```

```
    )
)
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.Dense(units=128, activation='relu'))
model.add(keras.layers.Dense(y_train.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['acc',f1_m,precision_m, recall_m])

history = model.fit(
    X_train, y_train,
    epochs=40,
    batch_size=64,
    validation_data=[X_test, y_test],
    shuffle=True,
    callbacks=[model_checkpoint_callback]
)

model.evaluate(X_test, y_test)
y_pred = model.predict(X_test)

from sklearn.metrics import confusion_matrix

def plot_cm(y_true, y_pred, class_names):
  cm = confusion_matrix(y_true, y_pred)
  fig, ax = plt.subplots(figsize=(18, 16))
  ax = sns.heatmap(
      cm,
      annot=True,
      fmt="d",
      cmap=sns.diverging_palette(220, 20, n=7),
      ax=ax
  )

  plt.ylabel('Actual')
  plt.xlabel('Predicted')
  ax.set_xticklabels(class_names)
  ax.set_yticklabels(class_names)
  b, t = plt.ylim()
  b += 0.5
  t -= 0.5
  plt.ylim(b, t)
```

```
  plt.show()

plot_cm(
  enc.inverse_transform(y_test),
  enc.inverse_transform(y_pred),
  enc.categories_[0]
)
```

Python код програмної реалізації класифікатора з GRU блоками

```
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from pandas.plotting import register_matplotlib_converters

%matplotlib inline
%config InlineBackend.figure_format='retina'

register_matplotlib_converters()
sns.set(style='whitegrid', palette='muted', font_scale=1.5)
rcParams['figure.figsize'] = 22, 10

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
tf.random.set_seed(RANDOM_SEED)

column_names = ['loggingTime', 'deviceID', 'batteryLevel', 'label',
'motionTimestamp_sinceReboot', 'motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ', 'motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ', 'groupCode']
df = pd.read_csv('grouped_united_data_v6.csv', header=0, names=column_names,
sep=';')
df.dropna(axis=0, how='any', inplace=True)

normalizedGroupedDf = df.sort_values(['groupCode','loggingTime',
'motionTimestamp_sinceReboot'], ascending=True).groupby('groupCode',
as_index=False, group_keys=False).apply(lambda x:
```

```
x.iloc[int(x.groupCode.size*0.1):int(x.groupCode.size*0.9)]).reset_index()
trainDf = normalizedGroupedDf.groupby('label', as_index=False,
group_keys=False).apply(lambda x: x.iloc[int(x.label.size *
0.2):int(x.label.size)]).reset_index()
testDf = normalizedGroupedDf.groupby('label', as_index=False,
group_keys=False).apply(lambda x: x.iloc[:int(x.label.size *
0.2)]).reset_index()

def plot_activity_acc(label, df):
    data = df[df['label'] == label][['motionUserAccelerationX',
'motionUserAccelerationY', 'motionUserAccelerationZ']][
           :1500]
    axis = data.plot(subplots=True, figsize=(16, 12),
                     title=label)
    for ax in axis:
        ax.legend(loc='lower left', bbox_to_anchor=(1.0, 0.5))


def plot_activity_rot(label, df):
    data = df[df['label'] == label][['motionRotationRateX',
'motionRotationRateY', 'motionRotationRateZ']][:1500]
    axis = data.plot(subplots=True, figsize=(16, 12),
                     title=label)
    for ax in axis:
        ax.legend(loc='lower left', bbox_to_anchor=(1.0, 0.5))

from sklearn.preprocessing import RobustScaler
scale_columns = ['motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ','motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ']
scaler = RobustScaler()
scaler = scaler.fit(trainDf[scale_columns].values)
trainDf.loc[:, scale_columns] =
scaler.transform(trainDf[scale_columns].to_numpy())
testDf.loc[:, scale_columns] =
scaler.transform(testDf[scale_columns].to_numpy())

from scipy import stats

def create_dataset(X, y, time_steps=1, step=1):
    Xs, ys = [], []
```

```
    for i in range(0, len(X) - time_steps, step):
        v = X.iloc[i:(i + time_steps)].values
        labels = y.iloc[i: i + time_steps]
        Xs.append(v)
        ys.append(stats.mode(labels)[0][0])
    return np.array(Xs), np.array(ys).reshape(-1, 1)

TIME_STEPS = 275
STEP = 25

X_train, y_train = create_dataset(
    trainDf[['motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ','motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ']],
    trainDf.label,
    TIME_STEPS,
    STEP
)

X_test, y_test = create_dataset(
    testDf[['motionUserAccelerationX', 'motionUserAccelerationY',
'motionUserAccelerationZ','motionRotationRateX', 'motionRotationRateY',
'motionRotationRateZ']],
    testDf.label,
    TIME_STEPS,
    STEP
)

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore', sparse=False)
enc = enc.fit(y_train)
y_train = enc.transform(y_train)
y_test = enc.transform(y_test)
from tensorflow import keras
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
```

```
def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))

def specificity(y_true, y_pred):
    neg_y_true = 1 - y_true
    neg_y_pred = 1 - y_pred
    fp = K.sum(neg_y_true * y_pred)
    tn = K.sum(neg_y_true * neg_y_pred)
    return tn / (tn + fp + K.epsilon())

checkPointPath =
'/Users/skrekoza/PycharmProjects/tensorflow_proj/gru_2_layer_best_test_v2/'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkPointPath,
    save_weights_only=True,
    monitor='f1_m',
    mode='max',
    save_best_only=True)

model = keras.Sequential()
model.add(
    keras.layers.Bidirectional(
      keras.layers.GRU(
          units=128,
          input_shape=[X_train.shape[1], X_train.shape[2]],
          return_sequences=True
      )
    )
)
model.add(keras.layers.Dropout(rate=0.2))
model.add(
    keras.layers.Bidirectional(
```

```
      keras.layers.GRU(
          units=64
      )
    )
)
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.Dense(units=128, activation='relu'))
model.add(keras.layers.Dense(y_train.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['acc',f1_m,precision_m, recall_m])
# model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.Adam(learning_rate=0.0005),
metrics=['acc',f1_m,precision_m, recall_m])
history = model.fit(
    X_train, y_train,
    epochs=40,
    batch_size=64,
    validation_data=[X_test, y_test],
    shuffle=True,
    callbacks=[model_checkpoint_callback]
)

model.evaluate(X_test, y_test)
y_pred = model.predict(X_test)

from sklearn.metrics import confusion_matrix

def plot_cm(y_true, y_pred, class_names):
  cm = confusion_matrix(y_true, y_pred)
  fig, ax = plt.subplots(figsize=(18, 16))
  ax = sns.heatmap(
      cm,
      annot=True,
      fmt="d",
      cmap=sns.diverging_palette(220, 20, n=7),
      ax=ax
  )

  plt.ylabel('Actual')
  plt.xlabel('Predicted')
  ax.set_xticklabels(class_names)
```

```
  ax.set_yticklabels(class_names)
  b, t = plt.ylim()
  b += 0.5
  t -= 0.5
  plt.ylim(b, t)
  plt.show()

plot_cm(
  enc.inverse_transform(y_test),
  enc.inverse_transform(y_pred),
  enc.categories_[0]
```

Частина Swift коду додатку для збору показників сенсорів, а саме класу CollectingDataVC

```
import UIKit
import CoreMotion
import CoreData
import WatchConnectivity


class CollectingDataVC: UIViewController, WCSessionDelegate, SettingsTableVCDelegate, RecordIDVCDelegate {


    // Statuses
    enum Status {
        case waiting
        case recording
    }

    var status: Status = Status.waiting {
        willSet(newStatus) {

            switch(newStatus) {
            case .waiting:
                print ("Stop recording on iPhone")
                waiting()
                break
```

```
        case .recording:
            print ("Start recording on iPhone")
            recording()
            break
        }
    }
    didSet {

    }
}


// Settings view controller
weak var settingsTableVC:SettingsTableVC?


// Controlls outlets
@IBOutlet weak var recordTimeLabel: UILabel!
@IBOutlet weak var recordStatusImage: UIImageView!
@IBOutlet weak var startButton: UIButton!
@IBOutlet weak var stopButton: UIButton!


// For session saving
var currentSession: Session? = nil
var nextSessionid: Int = 0
var recordTime: String = ""
var sensorOutputs = [SensorOutput]()
var characteristicsNames  = [CharacteristicName]()
var sessionType: SessionType = SessionType.OnlyPhone
var sensors = [Sensor]()


// Record stopwatch
var startTime = TimeInterval()
var UIUpdateTimer = Timer()


// Changing variable
var currentFrequency: Int = 0
var recordID: Int = 0
```

```
// For motion getting
let motion = CMMotionManager()
var motionUpdateTimer = Timer()


// MARK - Starting app

override func viewDidLoad() {
    super.viewDidLoad()

    //fillTestData()
    status = .waiting

    // Prepare for session
    if WCSession.isSupported() {
        let session = WCSession.default
        session.delegate = self
        session.activate()
    }

}

override func viewDidAppear(_ animated: Bool) {
    findLastSessionId()
    addNamesOfCharacteristics()
    addSensorIDs()
}


// MARK - connecting with delegates

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let destination = segue.destination
    if let settingsTableVC = destination as? SettingsTableVC {

        settingsTableVC.delegate = self
        self.settingsTableVC = segue.destination as? SettingsTableVC
    }
```

```
        if let recordIDVC = destination as? RecordIDVC {
            recordIDVC.delegate = self

            if let recordID = sender as? Int {
                recordIDVC.selectedID = recordID
            }
        }

    }



    // MARK - start / stop getting motion data

    func startGettingData() {

        // Make sure the motion hardware is available.
        if  self.motion.isAccelerometerAvailable,  self.motion.isGyroAvailable,
self.motion.isMagnetometerAvailable {

            self.motion.accelerometerUpdateInterval    =    1.0    /    Double
(currentFrequency)
            self.motion.gyroUpdateInterval = 1.0 / Double (currentFrequency)
            self.motion.magnetometerUpdateInterval    =    1.0    /    Double
(currentFrequency)

            self.motion.startAccelerometerUpdates()
            self.motion.startGyroUpdates()
            self.motion.startMagnetometerUpdates()

            // Configure a timer to fetch the data.
            self.motionUpdateTimer   =   Timer.scheduledTimer(withTimeInterval:
1.0/Double (currentFrequency), repeats: true, block: { (timer1) in
                // Get the motion data.
                if let dataAcc = self.motion.accelerometerData, let dataMag =
self.motion.magnetometerData, let dataGyro = self.motion.gyroData {

                    // let currenTime = self.returnCurrentTime()

                    let GyroX = dataGyro.rotationRate.x
                    let GyroY = dataGyro.rotationRate.y
```

```
                    let GyroZ = dataGyro.rotationRate.z

                    let AccX = dataAcc.acceleration.x
                    let AccY = dataAcc.acceleration.y
                    let AccZ = dataAcc.acceleration.z

                    let MagX = dataMag.magneticField.x
                    let MagY = dataMag.magneticField.y
                    let MagZ = dataMag.magneticField.z


                    //  print  (  "Gyro:  \(currenTime)  \(GyroX),  \(GyroY),
\(GyroZ)")
                    // print ( "Acc : \(currenTime) \(AccX), \(AccY), \(AccZ)")
                    // print ( "Mag : \(currenTime) \(MagX), \(MagY), \(MagZ)")


                    let sensorOutput = SensorOutput()

                    sensorOutput.timeStamp = Date()

                    sensorOutput.gyroX = GyroX
                    sensorOutput.gyroY = GyroY
                    sensorOutput.gyroZ = GyroZ

                    sensorOutput.accX = AccX
                    sensorOutput.accY = AccY
                    sensorOutput.accZ = AccZ

                    sensorOutput.magX = MagX
                    sensorOutput.magY = MagY
                    sensorOutput.magZ = MagZ

                    self.sensorOutputs.append(sensorOutput)

                }
            }
            )}
    }

    func stopGettingData() {
```

```
    motionUpdateTimer.invalidate()
    motionUpdateTimer = Timer()
    self.motion.stopGyroUpdates()
    self.motion.stopAccelerometerUpdates()
    self.motion.stopMagnetometerUpdates()
}

func returnCurrentTime() -> String {
    let date = Date()
    let calendar = Calendar.current
    let hour = calendar.component(.hour, from: date)
    let minutes = calendar.component(.minute, from: date)
    let seconds = calendar.component(.second, from: date)
    let nanoseconds = calendar.component(.nanosecond, from: date)

    let currentTime = "\(hour):\(minutes):\(seconds):\(nanoseconds)"

    return currentTime
}



// MARK - Delegate settings updates

func periodChangedNumberSettingsDelegate(_ number: Int){
    currentFrequency = number
}

func changeIDpressed(){
    performSegue(withIdentifier: "toRecordIDSettings", sender: recordID)
}

func recordIDChangedNumberSettingsDelegate(_ number: Int){
    recordID = number
    settingsTableVC?.recordID.text = "\(number)"
}



// MARK - Action controlls
```

```
    @IBAction func StartButtonpressed(_ sender: Any) {
        status = .recording

        startGettingData()
        UIUpdateTimer = Timer.scheduledTimer(timeInterval: 0.001, target: self,
selector: #selector(updateTime), userInfo: nil, repeats: true)
        startTime = NSDate.timeIntervalSinceReferenceDate

        // Start session recording
        currentSession = Session(context: context)
        currentSession?.id = Int32(nextSessionid)
        currentSession?.date = NSDate()
        currentSession?.frequency = Int32(currentFrequency)
        currentSession?.recordID = Int32(recordID)
        currentSession?.type = Int32(sessionType.rawValue)
    }


    @IBAction func stopButtonPressed(_ sender: Any) {

        // Finish session recording
        UIUpdateTimer.invalidate()
        currentSession?.duration = recordTime

        for sensorOutput in sensorOutputs {

            let characteristicGyro = Characteristic (context:context)
            characteristicGyro.x = sensorOutput.gyroX!
            characteristicGyro.y = sensorOutput.gyroY!
            characteristicGyro.z = sensorOutput.gyroZ!
            characteristicGyro.toCharacteristicName                          =
self.characteristicsNames[1]

            let characteristicAcc = Characteristic (context:context)
            characteristicAcc.x = sensorOutput.accX!
            characteristicAcc.y = sensorOutput.accY!
            characteristicAcc.z = sensorOutput.accZ!
            characteristicAcc.toCharacteristicName                           =
self.characteristicsNames[0]

            let characteristicMag = Characteristic (context:context)
```

```
            characteristicMag.x = sensorOutput.magX!
            characteristicMag.y = sensorOutput.magY!
            characteristicMag.z = sensorOutput.magZ!
            characteristicMag.toCharacteristicName                                =
self.characteristicsNames[2]


            let sensorData = SensorData(context: context)
            sensorData.timeStamp = sensorOutput.timeStamp! as NSDate
            sensorData.addToToCharacteristic(characteristicGyro)
            sensorData.addToToCharacteristic(characteristicAcc)
            sensorData.addToToCharacteristic(characteristicMag)
            sensorData.toSensor = sensors[0]

            self.currentSession?.addToToSensorData(sensorData)

        }


        if sessionType == SessionType.OnlyPhone {
            ad.saveContext()
            currentSession = nil
            sensorOutputs.removeAll()
        }

        print ("iPhone's motion data handled")
        nextSessionid += 1
        stopGettingData()
        status = .waiting
    }




    // MARK - Update UI Timer

    //Update Time Function
    @objc func updateTime() {
        let currentTime = NSDate.timeIntervalSinceReferenceDate

        //Find the difference between current time and start time.
```

```
        var elapsedTime: TimeInterval = currentTime - startTime

        //calculate the minutes in elapsed time.
        let minutes = UInt8(elapsedTime / 60.0)
        elapsedTime -= (TimeInterval(minutes) * 60)

        //calculate the seconds in elapsed time.
        let seconds = UInt8(elapsedTime)
        elapsedTime -= TimeInterval(seconds)

        //find out the fraction of milliseconds to be displayed.
        let fraction = UInt16(elapsedTime * 1000)

        //add the leading zero for minutes, seconds and millseconds and store
them as string constants
        let strMinutes = String(format: "%02d", minutes)
        let strSeconds = String(format: "%02d", seconds)
        let strFraction = String(format: "%03d", fraction)

        //concatenate minuts, seconds and milliseconds as assign it to the UILabel
        recordTimeLabel.text = "\(strMinutes):\(strSeconds):\(strFraction)"
        recordTime = "\(strMinutes):\(strSeconds)"
    }



    // MARK - Update changing state

    func waiting() {
        settingsTableVC?.periodSlider.isEnabled = true
        settingsTableVC?.currentRecordNumberLabel.text =  "\(nextSessionid)"
        settingsTableVC?.recordNumberLabel.text = "Next record number:"
        recordStatusImage.isHidden = true
        recordTimeLabel.isHidden = false
        recordTimeLabel.text = "00:00:000"
        startButton.isHidden = false
        stopButton.isHidden = false
        startButton.isEnabled = true
        stopButton.isEnabled = false
        settingsTableVC?.tableView.allowsSelection = true
    }
```

```
func recording() {
    settingsTableVC?.periodSlider.isEnabled = false
    recordStatusImage.isHidden = false
    settingsTableVC?.currentRecordNumberLabel.isHidden = false
    settingsTableVC?.currentRecordNumberLabel.text = "\(nextSessionid)"
    startButton.isEnabled = false
    stopButton.isEnabled = true
    settingsTableVC?.tableView.allowsSelection = false
    settingsTableVC?.recordNumberLabel.text = "Record number:"
}



//  MARK - Filling data for testing datamodel

func fillTestData(){

    let characteristicName1 = CharacteristicName (context:context)
    characteristicName1.name = "Gyro"
    let characteristicName2 = CharacteristicName (context:context)
    characteristicName2.name = "Acc"
    let characteristicName3 = CharacteristicName (context:context)
    characteristicName3.name = "Mag"



    let characteristic1 = Characteristic (context:context)
    characteristic1.x = 0.1
    characteristic1.y = 0.1
    characteristic1.z = 0.1
    characteristic1.toCharacteristicName = characteristicName1
    let characteristic2 = Characteristic (context:context)
    characteristic2.x = 0.2
    characteristic2.y = 0.2
    characteristic2.z = 0.2
    characteristic2.toCharacteristicName = characteristicName1
    let characteristic3 = Characteristic (context:context)
    characteristic3.x = 0.3
    characteristic3.y = 0.3
    characteristic3.z = 0.3
    characteristic3.toCharacteristicName = characteristicName1
```

```
let characteristic1a = Characteristic (context:context)
characteristic1a.x = 0.1
characteristic1a.y = 0.1
characteristic1a.z = 0.1
characteristic1a.toCharacteristicName = characteristicName2
let characteristic2a = Characteristic (context:context)
characteristic2a.x = 0.2
characteristic2a.y = 0.2
characteristic2a.z = 0.2
characteristic2a.toCharacteristicName = characteristicName2
let characteristic3a = Characteristic (context:context)
characteristic3a.x = 0.3
characteristic3a.y = 0.3
characteristic3a.z = 0.3
characteristic3a.toCharacteristicName = characteristicName2
let characteristic1b = Characteristic (context:context)
characteristic1b.x = 0.1
characteristic1b.y = 0.1
characteristic1b.z = 0.1
characteristic1b.toCharacteristicName = characteristicName3
let characteristic2b = Characteristic (context:context)
characteristic2b.x = 0.2
characteristic2b.y = 0.2
characteristic2b.z = 0.2
characteristic2b.toCharacteristicName = characteristicName3
let characteristic3b = Characteristic (context:context)
characteristic3b.x = 0.3
characteristic3b.y = 0.3
characteristic3b.z = 0.3
characteristic3b.toCharacteristicName = characteristicName3


let sensorData1 = SensorData(context: context)
sensorData1.timeStamp = NSDate()
sensorData1.addToToCharacteristic(characteristic1)
sensorData1.addToToCharacteristic(characteristic2)
sensorData1.addToToCharacteristic(characteristic3)

let sensorData2 = SensorData(context: context)
sensorData2.timeStamp = NSDate()
sensorData2.addToToCharacteristic(characteristic1a)
```

```
        sensorData2.addToToCharacteristic(characteristic2a)
        sensorData2.addToToCharacteristic(characteristic3a)

        let sensorData3 = SensorData(context: context)
        sensorData3.timeStamp = NSDate()
        sensorData3.addToToCharacteristic(characteristic1b)
        sensorData3.addToToCharacteristic(characteristic2b)
        sensorData3.addToToCharacteristic(characteristic3b)


        let session1 = Session(context: context)
        session1.id = 1
        session1.duration = "00:10"
        session1.date = NSDate()
        session1.frequency = 100
        session1.addToToSensorData(sensorData1)
        session1.addToToSensorData(sensorData2)
        session1.addToToSensorData(sensorData3)

        let session2 = Session(context: context)
        session2.id = 2
        session2.duration = "00:15"
        session2.date = NSDate()
        session2.frequency = 100
        session2.addToToSensorData(sensorData1)
        session2.addToToSensorData(sensorData2)
        session2.addToToSensorData(sensorData3)


        ad.saveContext()
    }



    //  MARK - Fetching and adding data to data model for local usage

    func findLastSessionId() {
        // Create Fetch Request
        let   fetchRequest   =   NSFetchRequest<NSFetchRequestResult>(entityName:
"Session")
        fetchRequest.fetchLimit = 1
```

```
        // Add Sort Descriptor
        let sortDescriptor = NSSortDescriptor(key: "id", ascending: false)
        fetchRequest.sortDescriptors = [sortDescriptor]

        do {
            let record = try context.fetch(fetchRequest) as! [Session]

            if record.count == 1 {
                let lastSession = record.first! as Session
                nextSessionid = Int(lastSession.id) + 1
                settingsTableVC?.currentRecordNumberLabel.text                =
"\(nextSessionid)"
            }

            else {
                nextSessionid = 0
                settingsTableVC?.currentRecordNumberLabel.text                =
"\(nextSessionid)"
            }

        } catch {
            print(error)
        }

    }

    func addNamesOfCharacteristics(){
        // Create Fetch Request
        let   fetchRequest   =   NSFetchRequest<NSFetchRequestResult>(entityName:
"CharacteristicName")

        // Add Sort Descriptor
        let sortDescriptor = NSSortDescriptor(key: "name", ascending: false)
        fetchRequest.sortDescriptors = [sortDescriptor]

        do {
            let     records     =     try     context.fetch(fetchRequest)     as!
[CharacteristicName]

            if records.count != 3 {
```

```
                let characteristicName1 = CharacteristicName (context:context)
                characteristicName1.name = "Gyro"
                let characteristicName2 = CharacteristicName (context:context)
                characteristicName2.name = "Acc"
                let characteristicName3 = CharacteristicName (context:context)
                characteristicName3.name = "Mag"
                ad.saveContext()
            }

        } catch {
            print(error)
        }

        // Populate local array
        let                              fetchRequestForLocalCharacteristicName:
NSFetchRequest<CharacteristicName> = CharacteristicName.fetchRequest()
        let  sortDescriptorForLocalCharacteristicName  =  NSSortDescriptor(key:
"name", ascending: true)
        fetchRequestForLocalCharacteristicName.sortDescriptors                =
[sortDescriptorForLocalCharacteristicName]
        do {
            self.characteristicsNames                    =                    try
context.fetch(fetchRequestForLocalCharacteristicName)
        }   catch   {
            print(error)
        }

    }

    func addSensorIDs(){
        // Create Fetch Request
        let   fetchRequest   =   NSFetchRequest<NSFetchRequestResult>(entityName:
"Sensor")

        // Add Sort Descriptor
        let sortDescriptor = NSSortDescriptor(key: "id", ascending: false)
        fetchRequest.sortDescriptors = [sortDescriptor]

        do {
            let records = try context.fetch(fetchRequest) as! [Sensor]
```

```
            if records.count != 2 {
                let sensor1 = Sensor(context: context)
                sensor1.id = 1
                let sensor2 = Sensor(context: context)
                sensor2.id = 2

                ad.saveContext()
            }

        } catch {
            print(error)
        }

        // Populate local array
        let      fetchRequestForLocalSensors:      NSFetchRequest<Sensor>      =
Sensor.fetchRequest()
        let   sortDescriptorForLocalSensors   =   NSSortDescriptor(key:   "id",
ascending: true)
        fetchRequestForLocalSensors.sortDescriptors                            =
[sortDescriptorForLocalSensors]
        do {
            self.sensors = try context.fetch(fetchRequestForLocalSensors)
        }   catch   {
            print(error)
        }

    }



    // MARK - Work with WCSessionDelegate

    // for receiving sessions
    func session(_ session: WCSession, didReceive file: WCSessionFile) {

        if sessionType == SessionType.OnlyPhone { return }

        print ("File with data received on iPhone!")
        print ("SessionType: \(sessionType)")

        let fm = FileManager.default
```

```
        let destURL = getDocumentsDirectory().appendingPathComponent("saved_file")

        do {
            if fm.fileExists(atPath: destURL.path) {

                // the file already exists - delete it
                try fm.removeItem (at: destURL)
            }

            // copy the file for its temporary location
            try fm.copyItem(at: file.fileURL, to: destURL)

            // load the file and print it out
            let mutableData = NSMutableData(contentsOf: destURL)

            let data = mutableData?.copy() as! Data

            let unarchiver = NSKeyedUnarchiver(forReadingWith: data)
            do {
                if let sessionContainerCopy = try unarchiver.decodeTopLevelDecodable(SessionContainer.self, forKey: NSKeyedArchiveRootObjectKey) {
                    // print("deserialized sensor output: \(String(describing: sessionContainerCopy.currentFrequency))")


                    DispatchQueue.main.async {
                        print("We are in main thread")

                        // print file size
                        let bcf = ByteCountFormatter()
                        bcf.allowedUnits = [.useMB] // optional: restricts the units to MB only
                        bcf.countStyle = .file
                        let string = bcf.string(fromByteCount: Int64(data.count))
                        print ("File size: \(string)")

                        // work with received data
                        print ("Start handling file...")
```

```
                        //                  sensorWatchOutputs                  =
sessionContainerCopy.sensorOutputs
                        if (self.sessionType == SessionType.PhoneAndWatch) {

                            for                  sensorOutput                  in
sessionContainerCopy.sensorOutputs {

                                let    characteristicGyro    =    Characteristic
(context:context)
                                characteristicGyro.x = sensorOutput.gyroX!
                                characteristicGyro.y = sensorOutput.gyroY!
                                characteristicGyro.z = sensorOutput.gyroZ!
                                characteristicGyro.toCharacteristicName       =
self.characteristicsNames[1]

                                let    characteristicAcc    =    Characteristic
(context:context)
                                characteristicAcc.x = sensorOutput.accX!
                                characteristicAcc.y = sensorOutput.accY!
                                characteristicAcc.z = sensorOutput.accZ!
                                characteristicAcc.toCharacteristicName        =
self.characteristicsNames[0]


                                let sensorData = SensorData(context: context)
                                sensorData.timeStamp = sensorOutput.timeStamp as
NSDate?
                                sensorData.toSensor = self.sensors[1]

sensorData.addToToCharacteristic(characteristicGyro)

sensorData.addToToCharacteristic(characteristicAcc)


self.currentSession?.addToToSensorData(sensorData)
                            }

                            print("Now starting saving to Data Core")
                            ad.saveContext()
                            print("After String saving")
                            self.sessionType = SessionType.OnlyPhone
```

```
                            self.sensorOutputs.removeAll()
                            // self.sensorWatchOutputs.removeAll()
                            self.currentSession = nil
                        }
                    }
                }
            } catch {
                print("unarchiving failure: \(error)")
            }

            print ("Finished saving into Data Core")


            let WCsession = WCSession.default
            if WCsession.activationState == .activated {
                let data = ["isFinishedHandling": true]
                WCsession.transferUserInfo(data)

                print("Sent callback to Watch")
            }

        }

        catch {
            // something went wrong
            print ("File copy failed")
        }

    }

    func getDocumentsDirectory() -> URL {

        let paths = FileManager.default.urls(for: .documentDirectory, in:
.userDomainMask)
        return paths[0]
    }

    // for receiving signals to start recording
    func session(_ session: WCSession, didReceiveMessage message: [String : Any],
replyHandler: @escaping ([String : Any]) -> Void) {
        DispatchQueue.main.async {
```

```
            if let isAlsoRun = message["Running"] as? Bool {

                if (isAlsoRun) {

                    if let recID = message["RecordID"] as? Int {
                        self.recordID = recID
                        self.settingsTableVC?.recordID.text = "\(recID)"
                    }

                    // updating frequency in all places
                    self.currentFrequency = 60
                    self.settingsTableVC?.periodSlider.setValue(60.0,  animated:
true)
                    self.settingsTableVC?.currentPeriodLabel.text = "60"

                    self.sessionType = SessionType.PhoneAndWatch
                    self.StartButtonpressed((Any).self)

                    // send back reply
                    replyHandler(["response": "Starting collecting data..."])

                } else {
                    self.stopButtonPressed((Any).self)

                    // send back reply
                    replyHandler(["response": "Stopping collecting data..."])
                }

            }
        }
    }

    func     session(_     session:     WCSession,     activationDidCompleteWith
activationState: WCSessionActivationState, error: Error?) {
        DispatchQueue.main.sync {
            if activationState == .activated {
                if session.isWatchAppInstalled {
                    print ("Watch app is installed")
                }
            }
```

```
        }
    }

    func sessionDidBecomeInactive(_ session: WCSession) {

    }

    func sessionDidDeactivate(_ session: WCSession) {
    }
}
```

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	96 с.
	Інші документи	
2.	Презентаційні матеріали	26 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Воронін Б.С			Інформаційно-аналітична система фітнес-клубу для визначення типу фізичної активності людини
Перевірив		Аврунін О.Г.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-21-2	122
ХНУРЕ кафедра ШІ	