МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВІДКРИТИЙ МІЖНАРОДНИЙ УНІВЕРСИТЕТ РОЗВИТКУ ЛЮДИНИ «УКРАЇНА»

Кваліфікаційна наукова

праця на правах рукопису

КИСЛИЙ РОМАН ВОЛОДИМИРОВИЧ

Гриф

Прим. № ____

УДК 004.42: 004.62: 004.72: 004.9:

ДИСЕРТАЦІЯ

РОЗПІЗНАВАННЯ ЛЮДСЬКИХ АКТИВНИХ ДІЙ ЗА ДОПОМОГОЮ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ

спеціальність 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Р.В. Кислий

Науковий керівник Петренко Анатолій Іванович, д.т.н., професор

Київ – 2021

АНОТАЦІЯ

Кислий Р. В. – Розпізнавання людських активних дій за допомогою методів штучного інтелекту – Рукопис.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 01.05.03 – математичне та програмне забезпечення обчислювальних машин і систем. – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2021.

Дисертація присвячена дослідженню методів розпізнавання активності людини та систем отримання даних з натільних датчиків у зв'язку з розвитком технологій Body Sensor Network. Такі технології дозволяють постійно моніторити фізіологічні показники людини, що в свою чергу дозволяє виявляти загострення хронічних хвороб, раптові кризові ситуації, а також здійснювати довготривалий моніторинг за здоров'ям людини.

Стрімкий ріст інтернету речей, загальна розповсюдженість інтернету створюють можливість для використання великої кількості сенсорів у повсякденному житті. При цьому сенсори можуть використовуватися для багатьох речей — від відслідковування якості повітря яким дихає людина до кількості кроків за день чи рівня цукру в крові. В умовах збільшення кількості населення та розповсюдження різних захворювань, системи охорони здоров'я рухаються у напрямку спостереження та зменшення контакту з пацієнтом (телемедицина). Однак такі процеси неможливі без даних про пацієнта, а відповідно і моніторингу його фізіологічних показників. Інтернет речей дозволяє за допомогою невеликих сенсорів збирати різні фізіологічні показники з людини і зразу показувати їх лікареві. Необхідність цифровізації медичних процесів також закріплено в Законі України «Про державні фінансові гарантії медичного обслуговування населення», де вказується необхідність побудови єдиної Електронної Системи Охорони Здоров'я (ЕСОЗ). Така система має мати захищені протоколи передачі даних, а також можливість добавляти сторонні сервіси як додаткові джерела даних. Цей підхід дозволяє розуміти що

відбувається з пацієнтом майже в реальному часі. Також, комплексні системи що збирають дані фізіологічного стану людини і аналізують дані в реальному часі мають змогу повідомляти про приступи чи нещасні випадки, що можуть трапитися з пацієнтом. Оскільки вони генерують величезну кількість даних, проводячи вимірювання в реальному часі, відповідно немає змісту переслати всі зняті дані до наступного рівня системи. Але, так як одна з основних функцій таких систем — реагувати на зміну чи загострення стану пацієнта, то для цього створюються моделі машинного навчання, які мають змогу розрізняти різні стани пацієнта за даними з сенсорів. Завдяки тому, що самі сенсори дуже енергоефективні, та мають постійний зв'язок з смартфоном, такі моделі можна розміщувати на смартфоні пацієнта, який проводить аналіз зібраних даних та здійснює зв'язок з наступним рівнем системи в разі необхідності.

Для підтримки такого підходу, в дисертаційній роботі запропоновано підходи для обробки даних з мережі натільних сенсорів, а також створення моделей для використання таких даних в визначенні поточного стану пацієнтів.

Для цього, задачу розпізнавання людської активності (HAR) розбито на різні підвиди, відповідно до типів сенсорів що використовуються для фіксації та вимірювання такої активності. Виділено окремо такі типи сенсорів, як натільні та такі, що не взаємодіють з людиною (наприклад відеокамера). Розглянута загальна структура HAR систем та розбита на компоненти:

- сенсорні технології
- ідентифікація діяльності
- відстеження діяльності
- класифікація діяльності.

В дисертації розглянуті різні види людської діяльності, та відповідно різні типи систем визначення такої діяльності. У відповідності до типу діяльності зроблено аналіз найбільш часто вживаних сенсорів для її фіксації. Акцент зроблено на використанні натільних сенсорів, так як вони можуть використовуватися в будь-

якому середовищі, без обмеження пересування людини. Як найбільш поширені сенсори для фіксації руху розглянуто акселерометри, гіроскопи, магнітометри, сенсори тиску та сферу їх застосування.

Проаналізовано структуру HAR систем, їх компонентів та класифікацію. Зокрема окрім сенсорів також розглянуто архітектуру систем HAR та етапи роботи з зібраними даними:

- процес збору даних
- обробки даних
- передача зібраних даних
- використання даних моделлю розпізнавання.

Значна увага приділяється методам оцінки точності систем HAR. Оскільки задача HAR відрізняється від простої задачі класифікації числових рядів тим, що кожна людина робить рухи з певними особливостями — різною швидкістю, амплітудою та іншими характеристиками, то і валідація моделей класифікації для задачі HAR відрізняється від стандартного методу валідації моделей машинного навчання. Для того щоб модель була корисною не тільки для однієї людини, записи з сенсорів збираються в однакових умовах для групи людей. Коли відбувається розділення зібраного набору даних на навчальний та тестовий, то до тестового повністю відносяться дані одного або кількох учасників тренувальної вибірки. Таким чином у валідаційному наборі даних модель отримує дані зібрані з людини, яка була відсутня в тренувальному наборі. Це дозволяє подивитися як змінюється точність моделі на даних, що мають особливості рухів, які модель раніше не бачила. В якості метрик точності моделей для класифікації HAR розглянуто універсальні метрики точності, повноти та F1 які дозволяють порівнювати між собою різні моделі.

Розглянуто найбільш популярні варіанти сенсорів та їх характеристики. До найбільш популярних з натільних сенсорів можна віднести акселерометри, гіроскопи та сенсори тиску. Хоча такі сенсори мають різні види виконання

(механічні, твердотільні і т.д.), найпопулярнішими є МЕМС (МікроЕлектроМеханічні), які мають значну перевагу, порівняно з іншими імплементаціями: малий розмір і вага (поміщається в смартфонах), низьке споживання енергії, висока надійність, дешевизна в виробництві, відсутність необхідності в обслуговуванні, майже відсутність вимог до середовища роботи.

Описано процес збору даних з натільних сенсорів. Особлива увага приділена архітектурі системи збору даних. Оскільки, перед тим як використовувати модель класифікації, необхідно її натренувати. Для цього збирається набір даних в контрольованому середовищі, які потім можна використати в якості тренування для навчання моделі.

Детально розглянута попередня обробка даних, зібраних з натільних сенсорів. Запропонований алгоритм очистки сирих даних для подальшого використання моделями класифікації. Проаналізовано використання різних методів знешумлення сигналу. Далі детально описано алгоритм визначення піків в сигналу з сенсора, що також служить в якості додаткової ознаки для подальшого використання в моделі машинного навчання класифікації дії.

Загальна архітектура такої системи складається з натільних сенсорів, що з’єднуються з мобільним пристроєм, використовуючи bluetooth, коли прилад є поблизу (в разі відсутності мобільного приладу в зоні досяжності, дані зберігаються на карті пам’яті). Після передачі даних вони проходять попередню обробку, і після цього класифікацію. Далі дані пересилаються в хмару, використовуючи API, де зберігаються в базу даних, і можуть бути в подальшому використаними для аналізу.

Традиційним підходом до обробки потоку даних, включаючи дані з сенсорів, що генеруються IoT, є використання лямбда-архітектури, яка об'єднує аналіз даних в режимі реального часу та історичних даних в рамках єдиної системи. Потоки даних надходять до черги повідомлень (або іншого джерела даних), де відбувається їх обробка:

1. За допомогою шару швидкісної обробки: Результати надаються в реальному часі синхронно (за допомогою відповідей на виклики API) або асинхронно (за допомогою спеціального API).
2. За допомогою шару пакетної обробки: необроблені дані зберігаються в озері даних, а потім обробляються та зберігаються в сховищі даних, і аналізуються за розкладом або на вимогу.

Для перевірки висунутих гіпотез був розроблений прилад, що складається з натільної частини, що кріпиться до грудної клітини: акселерометр, гіроскоп та сенсор тиску, карта пам’яті і модуль bluetooth. Такий прилад фіксує рух грудної клітини і записує покази на карту пам’яті. Якщо він підключений до мобільного пристрою за допомогою bluetooth, то також передає дані у розроблений додаток. Обробка даних відбувається на мобільному пристрою.

Щоб перевірити пропоновані рішення і гіпотези, було зібрано і сформовано набір даних. З цією метою було розроблено додаток для Android, який збирає дані з приладу, прикріпленого до грудей користувача, і передає дані на смартфон, використовуючи Bluetooth з частотою дискретизації 30 Гц (30 значень в секунду).

Щоб отримати розмічений набір даних, покази приладу було зібрано з 8 користувачів різного віку та фізичної форми. Для визначення різних типів дихання, користувачі перед вимірами робили відповідні рухи. Типи дихання, які записувалися включають: спокійне дихання, дихання після віджимання, дихання під час бігу, глибоке дихання, швидке дихання, дихання з затримкою. Використовуючи скотч-стрічку і пробуючи кілька різних місць на грудях, було емпірично з'ясовано, яку положення прилауд дає найбільшу амплітуду руху грудної клітини і, як наслідок, найчистіший сигнал дихання.

У роботі обґрунтовано методи збору та попередньої обробки даних з сенсорів. Для попередньої обробки таких даних розроблено алгоритм, який покращує фільтрацію шумів що виникають при зборі даних. Алгоритм передбачає перетворення одновимірні сигнали (1d) акселерометра в двовимірні (2d)

графічні зображення, за допомогою вікна Хеммінга, складаючи сигнали разом послідовно по рядках (stacking), що дозволяє кожній послідовності сигналів корелювати з іншими послідовностями. Доведено оптимальність такого підходу за допомогою порівняння з іншими і вимірюванні точності класифікації за допомогою моделей з однаковими архітектурами. Запропоновано архітектуру моделі на основі згорткової нейронної мережі, для ефективної класифікації людської діяльності, що складається з одного шару згортки, за яким слідує шар максимального об'єднання (pooling layer), і ще один шар згортки. Після цього модель містить повністю з'єднаний шар (fully-connected layer), який підключений до шару Softmax (багатолінійна логістична регресія).

Таким чином, поєднання вперше запропоновано комбінацію поєднання попередньої обробки даних у вигляді перетворення вхідного сигналу у 2d малюнок та використання згорткової нейронної мережі для класифікації типу людського дихання за допомогою даних з натільних сенсорів.

Ключові слова: машинне навчання, глибоке навчання, шаблони дихання, нейронні мережі, людська активність, обробка даних з сенсорів

ANNOTATION

Kyslyi R. V. – Human activity recognition using machine learning methods –Manuscript

Thesis for the degree of candidate of technical sciences. Specialty 01.05.03 − "Mathematical maintenance and software for computers and systems".

- National Technical University of Ukraine "Kyiv Polytechnic Institute named after Igor Sikorsky", Kyiv, 2021.

The dissertation is focused on research of methods of recognition of human activities and systems of data gathering from wearable sensors regarding the development of Body Sensor Network technology. Such technologies allow to constantly monitor physiological parameters of the person and allow to reveal exacerbations of chronic diseases, sudden crisis situations, and also provide long-term monitoring for human health.

The rapid growth of the Internet of Things, the general prevalence of the Internet creates opportunities to use a large number of sensors in everyday life. At the same time, sensors can be used for many things - from monitoring the air quality a person breathes to the number of steps per day or blood sugar levels. With an increasing population and the spread of various diseases, health care systems are moving in the direction of monitoring and reducing contact with the patient (using telemedicine). However, such processes are impossible without gathering data about the patient and, accordingly, monitoring of his physiological parameters. The Internet of Things allows you to use small sensors to collect various physiological parameters from a person and immediately deliver them to the doctor. The need for digitalization of medical processes is also enshrined in the Law of Ukraine "On State Financial Guarantees of Medical Care", which indicates the need to build a single Electronic Health Care System (EHS). Such a system must have secure data transfer protocols, as well as the ability to add third-party services as additional data sources. This approach allows you to understand what is happening to the patient in almost real time. Also, complex systems that collect data on the physiological state of a person and analyse data in real time have the ability to report critical conditions or accidents that may occur to the patient. Since they generate a huge amount of data by performing real-time measurements, it makes no sense to forward all the captured data to the next level of the system. However, since one of the main functions of such systems is to respond to changes or exacerbations of the patient's condition, machine learning models are created for this purpose, which are able to distinguish different patient conditions based on sensor data. Due to the fact that the sensors themselves are very energy efficient and have a constant connection to the smartphone, such models can be placed on the smartphone of the patient, which analyses the collected data and communicates with the next level of the system if necessary.

To support this approach, in the dissertation are proposed approaches for data processing from a network of body sensors, as well as the creation of models for the use of such data in determining the current status of patients.

To do this, the task of recognizing human activity (HAR) is divided into different subtypes, according to the types of sensors used to record and measure such activity. There are separate types of sensors, such as body and those that do not interact with humans (such as a video camera). The general structure of HAR systems is considered and divided into components:

- sensory technologies
- activity identification
- activity tracking
- activity classification

The dissertation considers of research of different types of human activity, and accordingly different types of systems for determining such activity. In accordance with the type of activity, the analysis of the most frequently used sensors for its fixation is made. The emphasis is on the use of body wearn sensors, as they can be used in any environment, without restricting human movement. Accelerometers, gyroscopes, magnetometers, pressure sensors and their scope are considered as the most common sensors for motion detection.

The structure of HAR systems, their components and classification are analysed. In particular, in addition to sensors, the architecture of HAR systems and stages of work with the collected data are also considered:

• data collection process

• data processing

• transfer of collected data

• use of data recognition model.

In the dissertation, much attention is paid to the methods of the accuracy of HAR systems. Since the HAR problem differs from the simple problem of classifying numerical time series, in that each person makes movements with certain features - different speed, amplitude and other characteristics, the validation of classification models for

the HAR problem differs from the standard method of validation of machine learning models. In order for the model to be useful for more than one person, sensor recordings are collected under the same conditions for a group of people. When the collected data set is divided into training and test set, the test data set fully includes the data of one or more participants in the training sample. Thus, in the validation data set, the model receives data collected from a person who was not included in the training set. This allows you to see how the accuracy of the model changes on data that have features of movements that the model has not seen before. Universal metrics of accuracy, recall and F1 which allow to compare different models among themselves are considered as metrics of accuracy of models for HAR classification.

The most popular variants of sensors and their characteristics are reviewed. Including accelerometers, gyroscopes and pressure sensors. Although such sensors have different types of performance (mechanical, solid-state, etc.), the most popular are MEMS (MicroElectroMechanical), which have a significant advantage over other implementations: small size and weight (placed in smartphones), low power consumption, high reliability, cheapness in production, no need for maintenance, almost no requirements for the work environment.

In the dissertation described the process of data collection from the body sensors. Particular attention is paid to the architecture of the data collection system. Because, before using the classification model, it is necessary to train it. To do this, a set of data is collected in a controlled environment, which can then be used as training to train the model.

In detail reviewed pre-processing of data collected from body sensors. An algorithm for cleaning raw data for further use by classification models is proposed. The use of different methods of signal noise reduction is analyzed. The following describes in detail the algorithm for determining the peaks in the signal from the sensor, which also serves as an additional feature for further use in the model of machine learning action classification.

The general architecture of such a system consists of body sensors that connect to a mobile device using bluetooth when the device is nearby (in the absence of a mobile device within range, the data is stored on a memory card). After data transfer, they are pre-processed, and then classified. Then the data is sent to the cloud using the API, where it is stored in the database, and can be further used for analysis.

The traditional approach to processing data flow, including data from sensors generated by IoT, is to use a lambda architecture that combines real-time data analysis and historical data within a single system. Data streams enter the message queue (or other data source), where they are processed:

3. Using the speed processing layer: The results are provided in real time synchronously (using responses to API calls) or asynchronously (using a special API).
4. 3. Using a batch processing layer: raw data is stored in a data lake and then processed and stored in a data warehouse, and analyzed on a schedule or on demand.

To test the hypotheses proposed in the dissertation, a device consisting of a body attached to the chest was developed: an accelerometer, a gyroscope and a pressure sensor, a memory card and a bluetooth module. This device records the movement of the chest and records the readings on a memory card. If it is connected to a mobile device via bluetooth, it also transmits data to the developed application. Data processing takes place on a mobile device.

To test the proposed solutions and hypotheses, a data set was collected and generated. To this end, an Android application has been developed that collects data from a device attached to the user's chest and transmits data to a smartphone using Bluetooth with a sampling rate of 30 Hz (30 values per second).

To get a marked data set, the device readings were collected from 8 users of different ages and fitness. To determine the different types of breathing, users made appropriate movements before measuring. Types of breathing that were recorded include: calm breathing, breathing after squeezing, breathing while running, deep breathing, rapid

breathing, breathing with delay. Using scotch tape and trying several different places on the chest, it was empirically determined which position of the device gives the greatest amplitude of movement of the chest and, consequently, the purest signal of respiration.

The methods of data collection and pre-processing from sensors are substantiated in the work. To pre-process such data, an algorithm has been developed that improves the filtering of noise arising during data collection. The algorithm involves converting one-dimensional signals (1d) of the accelerometer into two-dimensional (2d) graphics, using a Hamming window, stacking the signals together sequentially in rows (stacking), which allows each sequence of signals to correlate with other sequences. The optimality of this approach is proved by comparing with others and measuring the accuracy of classification using models with the same architectures. A model architecture based on a convolutional neural network is proposed for the effective classification of human activity, which consists of one convolution layer, followed by a pooling layer, and another convolution layer. The model then contains a fully-connected layer, which is connected to the Softmax layer (multi-line logistic regression).

Thus, the combination is the first proposed combination of pre-processing data in the form of converting the input signal into a 2d image and using a convolutional neural network to classify the type of human respiration using data from body sensors.

Keywords: machine learning, deep learning, breathing patterns, neural networks, human activity, sensor data processing

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових фахових виданнях України

1. Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018). Detection of human respiration patterns using deep convolution neural networks. Eastern-European Journal of

Enterprise Technologies, 4(9 (94)), 6–13. https://doi.org/10.15587/1729-4061.2018.139997

2. Roman V. Kyslyi, Anatolii I. Petrenko,/ (2020) Розпізнавання людської діяльності за допомогою портативних натільних датчиків, Системні дослідження та інформаційні технології, no. 2,pp. 41-54 2020, doi:https://doi.org/10.20535/SRIT.2308-8893.2020.2.03

3. Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018). Designing security of personal data in distributed health care platform. Technology Audit and Production Reserves, 2(42). https://doi.org/10.15587/2312-8372.2018.141299

4. Pysmennyi, I., Kyslyi, R., & Petrenko, A. (2019). Edge computing in multi-scope service-oriented mobile healthcare systems. System Research and Information Technologies, 0(1), 118–127. https://doi.org/10.20535/SRIT.2308-8893.2019.1.09

Статті в закордонних виданнях

5. I. Pysmennyi, A. Petrenko, and R. Kyslyi, (2020) Graph-based fog computing network model, Applied Computer Science, vol. 16, no. 4, pp. 5-20, 2020, doi: 10.23743/acs-2020-25

Зміст

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HAR – розпізнавання людської діяльності, human activity recognition

CNN – згорткова нейронна мережа, convolution neural network

RNN – рекурентна нейронна мережа, recurrent neural network

LSTM - довга короткочасна пам'ять, long short term memory

FFT – швидке перетворення Фур’є, fast Fourier transform

DFT – дискретне перетворення Фур’є, discrete Fourier transform

TCN - часові згорткові мережі, temporal convolution networks

MEMC - МікроЕлектроМеханічні системи

АЧХ – амплітудно-частотна характеристика

PCA - метод головних компонент, principal component analysis

СОА – сервісно орієнтована архітектура

MSA – мікросервісна архітектура, microservice architecture

Вступ

Актуальність теми дослідження. Стрімкий ріст інтернету речей, загальна розповсюдженість інтернету створюють можливість для використання великої кількості сенсорів у повсякденному житті. При цьому сенсори можуть використовуватися для багатьох речей — від відслідковування якості повітря яким дихає людина до кількості кроків за день чи рівня цукру в крові. В умовах збільшення кількості населення та розповсюдження різних захворювань, системи охорони здоров'я рухаються у напрямку спостереження та зменшення контакту з пацієнтом (телемедицина). Однак такі процеси неможливі без даних про пацієнта, а відповідно і моніторингу його фізіологічних показників. Інтернет речей дозволяє за допомогою невеликих сенсорів збирати різні фізіологічні показники з людини і зразу показувати їх лікареві. Необхідність оцифровування медичних процесів також закріплено в Законі України «Про державні фінансові гарантії медичного обслуговування населення», де вказується необхідність побудови єдиної Електронної Системи Охорони Здоров'я (ЕСОЗ). Така система має мати захищені протоколи передачі даних, а також можливість додавати сторонні сервіси як додаткові джерела даних. Такий підхід дозволяє розуміти що відбувається з пацієнтом майже в реальному часі. Також, комплексні системи що збирають дані фізіологічного стану людини і аналізують дані в реальному часі мають змогу повідомляти про приступи чи нещасні випадки, що можуть трапитися з пацієнтом. Оскільки вони генерують величезну кількість даних, проводячи вимірювання в реальному часі, відповідно немає змісту пересилати всі зняті дані до наступного рівня системи. Але, так як одна з основних функцій таких систем — реагувати на зміну чи загострення стану пацієнта, то для цього створюються моделі машинного навчання, які мають змогу розрізняти різні стани пацієнта за даними з сенсорів. Завдяки тому, що самі сенсори дуже енергоефективні, та мають постійний зв'язок з смартфоном, такі моделі можна розміщувати на смартфоні пацієнта,

який проводить аналіз зібраних даних та здійснює зв’язок з наступним рівнем системи в разі необхідності.

Для підтримки даних компонентів в дисертаційній роботі запропоновано підходи для обробки даних з мережі натільних датчиків, а також створення моделей для використання таких даних у визначенні поточного стану пацієнтів.

Зв'язок роботи з науковими програмами, планами, темами

Дисертаційна робота пов’язана а з планами наукових досліджень, які виконувалися на кафедрі системного проектування КПІ ім. Ігоря Сікорського з розвитком теоретичної бази нової спеціалізації з системного проектування сервісів. Результати роботи використовувалися при виконанні НДР №2020-п «Проектування сучасних систем сервісів на прикладі мобільної медичної системи для мешканців прифронтових районів в зоні АТО» (номер державної реєстрації 0117U002435, терміни виконання 2017-2020 рр.), у яких здобувач приймав участь як виконавець.

Мета і задачі дисертаційного дослідження. Метою дисертаційної роботи є підвищення ефективності дослідження людської активності, зокрема моніторингу за станом здоров’я.

Відповідно до поставленої мети необхідно розв’язати наступні наукові задачі:

- здійснити аналіз існуючих методів, моделей та алгоритмів що використовуються для визначення людської активності;
- розробити загальну архітектуру та підхід для систем визначення людської активності;
- для попереднього опрацювання вхідних даних розробити та обґрунтувати оптимальний за складністю алгоритм обробки вхідних даних з натільних сенсорів для подальшого використання в моделі класифікації людської діяльності;

- розробити пристрій для зняття фізіологічних даних для збору набору даних;
- розробити та дослідити доцільну архітектуру моделі машинного навчання для конкретної задачі визначення людської діяльності;
- дослідити та використати моделі, що навчені на інших наборах даних, щодо можливості їх використання для класифікації людської діяльності використовуючи трансферне навчання.

Об'єкт дослідження: процес визначення типу та якості виду людської діяльності.

Предмет дослідження: методи обробки даних з натільних датчиків, архітектура моделей машинного навчання для класифікації людської діяльності

Методи дослідження: Для досягнення поставленої мети в дисертаційній роботі використано низку теорій та методів. **Зокрема,** для обробки та аналізу даних з натільних датчиків, використані методи цифрової обробки сигналів, методи системного аналізу, методи агрегації, теорія алгоритмів. Розробка архітектур моделей машинного навчання здійснювалась з використанням теорії алгоритмів та математичної статистики.

Наукова новизна отриманих результатів.

В дисертації вперше одержані такі нові наукові результати:

- Проведено аналіз існуючих систем розпізнавання дій людини, який показав, що певні недоліки і незручності використання портативних натільних датчиків низького рівня можна компенсувати за рахунок методів машинного навчання і тим забезпечити багату контекстуальну інформацію в реальному застосуванні. Тому на відміну від існуючих прототипів в роботі наголос зроблено на дослідження властивостей таких систем при використанні новітніх згорткових нейронних мереж (CNN).

- Вперше запропоновано для підвищення ефективності застосування згорткової нейронної мережі (CNN) перетворювати одновимірні сигнали (1d) акселерометра в двовимірні (2d) графічні зображення , які обробляються за допомогою CNN із декількома обробними шарами, завдяки чому точність визначення типу дії людини в різних ситуаціях для різних фізичних станів зростає в порівнянні з випадком, коли двовимірні перетворення сигналів акселерометра не вживаються.
- Вперше для оброблення сигналів датчика-акселерометра в умовах наявності перешкоджаючих сигналів (шумів) від інших джерел та інструментальних похибок пристрою запропоновано перетворювати сигнал з допомогою вікна Хеммінга, складаючи сигнали разом послідовно по рядках (stacking) у вигляді зображення, що дозволяє кожній послідовності сигналів корелювати з іншими послідовностями.
- Вперше висунута і перевірена гіпотеза про можливість застосування систем розпізнавання дій людини для моніторингу руху грудної клітини, через який визначається частота і глибина вдихів при диханні, притаманні певним хворобам, включаючи сонне апное (зупинка дихання під час сну).

Практичне значення отриманих результатів

Практична цінність результатів роботи полягає в наступному:

- Проведене теоретичне обґрунтування, розробка системи що забезпечує моніторинг дихання людини в реальному часі, побудовано моделі, які забезпечують розпізнавання зупинку дихання, що дозволяє вчасно подати сигнал людині уві сні. Це певний внесок даної роботи в можливості моніторингу за станом власного здоров’я.
- Розроблені методи попередньої обробки даних, що надходять з натільних датчиків. Такі методи дозволяють покращити точність обробки даних, і як

результат покращити точність передбачення чи аналізу систем, що використовують дані з натільних датчиків.

- Розроблена архітектура системи з елементами машинного навчання, що має змогу спостерігати за показниками життєдіяльності людини чи її діями в реальному часі, при цьому розміщуючись на мобільних платформах, що дозволяє використовувати таку систему на протязі тривалого часу без великих елементів живлення.
- Запропоновано підхід до трансферного навчання в задачі визначення людської діяльності, що дозволяє використовувати попередньо натреновані моделі для інших задач.

Особистий внесок здобувача в отриманні наукових та практичних результатів, що викладені в дисертаційній роботі

Дисертаційна робота є узагальненням результатів теоретичних і експериментальних досліджень, проведених автором самостійно. У базових роботах [1,3], дисертанту належать: аналіз методу збору даних з натільних датчиків, розробка методів попередньої обробки даних, розробка моделей машинного навчання для виявлення людської активності. У роботі [2] — аналіз роботи моделей машинного навчання в децентралізованих розподілених мережах; [4] – розвиток методів збереження персональних даних пацієнтів в розподілених мережах; [5] – аналіз сервісного підходу до проектування медичних мобільних сервісів;

Дисертаційна робота виконана на кафедрі системного проектування під керівництвом зав. кафедри СП, д.т.н., проф. Петренко А. І.

Робота є результатом самостійних досліджень Кислого Р.В.

Апробація результатів дисертації

Результати досліджень, які включено в дисертацію, були представлені на міжнародному форумі Innovation Market та Китайсько-Українській науковій виставці технологій та інновацій з 21 по 24 листопада 2017р., конкурсі стартап-

проектів Всеукраїнського фестивалю інновацій 15-17 травня 2018 року з проектом “AirMonitor” (півфіналісти), та з проектом “Виявлення шаблонів дихання за допомогою глибоких згорткових нейронних мереж (CNN)”, проект переміг на конкурсі стартапів “SIKORSKY CHALLENGE 2018”.

Розділ 1 Стан розробок систем розпізнавання людської діяльності

1.1 Визначення проблеми human activity recognition

Підходи до визначення виду людської діяльності (human activity recognition (далі HAR)), можна традиційно розділити на такі, що використовують натільні датчики, і такі, що використовують зовнішні датчики (тобто такі, що не кріпляться до тіла людини: наприклад, камери). В обох випадках, дані з тих, чи інших датчиків збирають в часові ряди, що дозволяє привести визначення задачі розпізнавання людської активності [1]:

Визначення 1. Загальне визначення HAR:

Якщо дано набір вимірів S, такий що S={S0, ..., Sk−1}, де k — кількість вимірів, кожен з яких зроблений в інтервалі від I = [tα, tω], то ціль — знайти відповідне I з набору значень станів I=[I_0, ... , I_r], базуючись на даних з S. При чому, в кожний момент часу і, може розглядатися лише один стан I (наприклад людина не біжить і не сидить одночасно) [1].

Враховуючи, що можлива майже безкінечна кількість комбінацій різних рухів, які людина може зробити одночасно, то визначити момент, коли одна активність перетікає в іншу буває дуже складно, а інколи і неможливо. Таким чином, для того, щоб алгоритм мав можливість розпізнати активність, записаний часовий ряд розбивають на вікна з фіксованою довжиною. Звідси виходить друге визначення задачі розпізнавання людської активності:

Визначення 2 (проблема з розслабленою HAR):

Дано множину $W = \{W_0, ..., W_{m-1}\}$ m однакових за розміром часових вікон, повністю або частково маркованих певною активністю. Кожне W_i містить в собі набір часових рядів $S_i = \{S_i, 0, ..., S_i, k - 1\}$ від кожного з k виміряних атрибутів, і множину $A = \{a_0, ..., a_{n-1}\}$ міток активності. Мета полягає у пошуку функції відображення $f: S_i \rightarrow A$, яку можна оцінити для всіх можливих значень S_i, таким чином, щоб $f(S_i)$ був максимально схожим на фактичну активність, що виконується під час W_i. [2]

Таким чином, виходячи з цих визначень, складові конкретної системи HAR залежать в першу чергу від активності, яку необхідно визначити. Зміна цієї активності може викликати зміну необхідних датчиків, і методів обробки їх даних. Тим не менше, всі системи HAR в загальному мають схожу структуру.

1.2 Загальна структура HAR систем

Таким чином у моделі HAR для взаємодії людини і складного комп'ютеризованого агрегату є чотири основні технічні компоненти:

- сенсорні технології
- ідентифікація людської діяльності
- відстеження людської діяльності
- класифікація людської діяльності.

Відповідно до визначення, HAR включає в себе вивчення рухів людини. Рух людини - це складна функція, на яку впливає безліч факторів, включаючи фізіологічні, анатомічні, психологічні та соціальні ефекти [3], тому самі рухи можна розбити на великі підгрупи (Табл. 1).

Таблиця 1. Типи активності, які можуть розпізнавати найсучасніші HAR системи

Група	Активності
Пересування	Ходьба, біг, сидіння, лежання, підйом сходами, спуск сходами, їзда на ліфті, тощо.
Рутинні активності	Їжа, пиття, робота за ПК, читання, чистка зубів, розтягування, прибирання, тощо.
Вправи	Веслування, підняття ваги, нордична ходьба, віджимання, тощо.

Різні рухи мають різну складність та точність розпізнавання. Наприклад, ходьбу, біг чи спокій простіше відрізнити ніж складніші рухи на зразок жування, вставання з місця чи конкретного помаху рукою.

Для розпізнавання цих рухів використовують системи натільних датчиків, які (можуть кріпитися на поясі, на грудях, на зап’ясті та інших місцях) кріпляться в різних місцях: на поясі, на грудях, на зап’ясті, і інше.

Відповідно в різних системах використовуються різні сенсори. Найчастіше використовують:

- *Акселерометри* - використовують для вимірювання прискорення вздовж осей. Оскільки вони вимірюють прискорення за рахунок сили тяжіння та руху, фактичну складову прискорення, пов'язаного з рухом, потрібно відокремити від гравітаційної складової . Існує кілька типів акселерометрів, що базуються на п'єзоелектричних, п’єзорезистивних або варіативних способах трансдукції. Усі вони використовують один і той же принцип роботи маси, яка реагує на прискорення,
- *Гіроскопи* — вимірюють кутову швидкість, використовуючи ефект Коріоліса.
- *Магнітометри* - можуть бути використані для вимірювання орієнтації відрізка тіла щодо магнітного полюсу Землі, використовуючи електромагнітну індукцію.
- *Датчики тиску* - оцінюють розподіл тиску на планку (наприклад стопи, чи іншої частини тіла). Вони часто реалізуються за допомогою резистивних або ємнісних тензодатчиків.
- *Актометри* - зазвичай прикріплюють до кінцівок людини, щоб виміряти величину механічних рухів. Отриманий вихід є вимірюванням "одиниць актометра" за відомий проміжок часу, що дає змогу оцінити загальні витрати енергії [3].

Розташування датчиків на корпусі сильно залежить від того, які рухи необхідно вимірювати. Наприклад, зап'ястя може бути ідеальним місцем для спостереження за тремором, пов’язаним із хворобою Паркінсона, але це не

найкраще місце для вимірювання загальних рухів людини [4]. В той самий час, акселерометр закріплений на поясі, добре вимірює загальні рухи (наприклад ходьбу чи біг), і зовсім не підходить для спостереження за хворобою Паркінсона.

Також, велика частина способів, спрямованих на HAR, застосовує техніки комп'ютерного зору і обробки зображень [3] [4]. Для цього необхідна наявність камер і присутність людини у видимій зоні. Розвиток мобільних технологій і вдосконалення технологічних процесів ведуть до зменшення розмірів і збільшення потужності мобільних пристроїв, появи переносної електроніки, оснащеної різними датчиками.

1.3 Процес HAR

Проектування будь-якої системи HAR, в першу чергу, залежить від типів активності, які необхідно розпізнати. При зміні множини станів, зразу виникає необхідність в перетренуванні моделей розпізнавання [5] Тому будь-який проект HAR складається з таких стадій [6]:

- визначення типів активності , які мають бути розпізнані;
- збір даних з сенсорів;
- початкова обробка даних та генерація ознак;
- створення моделі класифікації діяльності.

Рисунок 1.1. Загальна діаграма процесу в HAR системі

Рис.1.1 ілюструє стадії функціонування HAR системи. Навчальний етап спочатку вимагає збору даних, що включає в себе створення набору даних з часових рядів, для всіх осіб, і для кожної діяльності, що вимірюють. Часові ряди розділяють, використовуючи вікна, щоб дістати ознаки (наприклад спектральну складову, частотну характеристику, тощо), тим самим відфільтровуючи корисну інформацію з початкових сигналів. Після цього, створений набір даних використовують для навчання моделей розпізнавання типів активності. Так само для тестування дані збирають під час часового вікна з окремої людини, виміри даних з якої не входять в навчальний набір даних. (Так само при тестуванні окремої людини, вимірювані дані збирають під час часового вікна і ці виміряні дані не входять в навчальний набір даних)

Загальна схема збору даних для систем HAR виглядає так, як зображено на Рис.1.2.

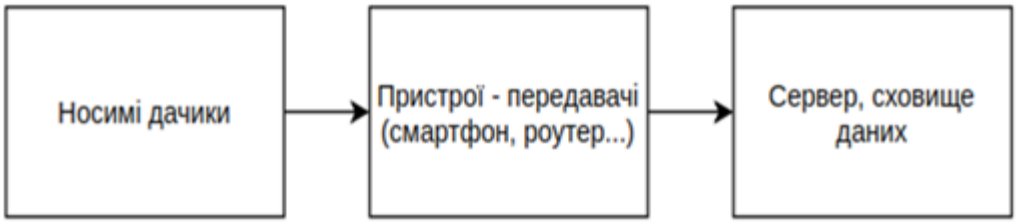


Рисунок 1.2. Загальна схема збору даних для HAR

В першу чергу натільні датчики, прикріплені до тіла людини, вимірюють такі атрибути, як рух [4], розташування, температура, ЕКГ [7], та інші. Ці датчики повинні взаємодіяти з інтеграційним пристроєм (ID), який може бути мобільним телефоном [8], ноутбуком або спеціалізованою вбудованою системою [9]. Основна мета - попередня обробка даних, отриманих від датчиків, а в деяких випадках також надсилання їх на сервер для моніторингу, візуалізації та / або аналізу в режимі реального часу [8]. Протокол зв'язку може бути UDP / IP або TCP / IP, GPRC, чи інший, відповідно до бажаного рівня надійності [10].

1.4 Обробка даних, отриманих з датчиків

Враховуючи, що дані, які надходять з датчиків є значеннями часового ряду, то задачу HAR для попередньої обробки даних, можна сприймати як задачу класифікації часових рядів [4]. Таким чином постає задача попередньої обробки даних перед подачею їх в модель класифікації. Згідно з [1], для обробки даних часових рядів перед класифікацією найбільш вживані — наступні методи:

- Зведення сигналу до однієї довжини — додавання нулів, або використання медіанного значення
- Дискретне перетворення Фур’є [11]
- Виділення різниці — видалення тренду (для прикладу, тренд можна видалити, віднявши попереднє значення від кожного наступного значення в ряду)

- Стандартизація — перетворення розподілу даних на нормальний (від кожного значення в ряді віднімається середнє значення і результат ділиться на стандартне відхилення)
- Нормалізація — зміна діапазону даних з початкового до нового, між 0 і 1.

Також, враховуючи що дані збирають з датчиків, необхідно враховувати потенційну наявність аномальних значень, оскільки неможливо гарантувати постійну якість сигналу з зовнішньої системи. Виявлення аномалії для часових рядів можна сформулювати, як пошук зовнішніх точок даних щодо якогось стандартного або звичайного значення сигналу. В числових рядах виділяють 3 основні типи аномалій: глобальні аномалії (значення, які набагато більші або менші відносно всіх інших даних в ряді), контекстні аномалії (значення значно відрізняється від інших значень в контексті — наприклад, значення, отримане з датчика, значно більше, ніж всі інші значення, попередньо отримані з даного датчика), колективні аномалії (наприклад кореляція значень кількох датчиків значно відрізняється від їх кореляції в інші моменти часу, але при цьому, окремі значення датчиків не є аномальними) [12].

Такі значення можуть виникати через різні причини - несправність датчика, спотворення даних при передачі. Методи визначення аномалій достатньо складні. Часто вони вимагають побудови окремої моделі, і базуються на передбаченні наступного значення ряду в довірчому інтервалі. В [13] описуються методи відсіювання аномалій для часових рядів:

- Класифікаційні та регресивні дерева — модель передбачає наступну точку ряду. Для перевірки, чи точка потрапляє в довірчий інтервал, використовується критерій Граббса (при цьому ряд має мати нормальний розподіл) [14]. Такий підхід ніяк не пов’язаний з структурою даних, що дозволяє враховувати багато параметрів. З іншого боку, це рішення не використовується в системах реального часу, бо вимагає досить екстенсивних обчислень.

- ARIMA (Autoregressive integrated moving average) — підхід передбачає прогнозування наступної точки ряду з використанням кількох попередніх і додавання певного шуму [15]. Однак, хоч цей метод і простіший з точки зору обчислень, він вимагає побудови нової моделі для кожного нового сигналу.
- Нейронні мережі — використання різних архітектур нейронних мереж для знаходження аномалій. [16]

1.5 Методи оцінки точності системи HAR

Для підвищення точності даних необхідно зібрати як можна більше вимірювань з різних людей. При цьому можуть використовуватися різні датчики, які записують свої показання синхронно і одночасно.

Як описувалося раніше, з первинних даних необхідно зробити аналіз ознак і виділити ті з них, які допомагають в класифікації активності. Обробка зібраних даних, так само як і генерація ознак, відбувається окремо для кожного датчика. Наприклад, для акселерометра, який вимірює прискорення вздовж трьох осей, покази трансформуються в одне значення a:

$$a = \sqrt{x^2 + y^2 + z^2}$$

Так само, нормалізуються також і показання інших датчиків. Також при генерації ознак використовують і багато інших технік і метрик Наприклад Редді та інші [17] використовують середні, дисперсійні, енергетичні коефіцієнти дискретної трансформації Фур'є (DFT). Ці всі нові ознаки, які створюються під час обробки початкових даних, потім використовуються в моделях класифікації. Оскільки кожен набір даних демонструє різні характеристики та ознаки, які можуть бути корисними, або ні, для конкретного методу розпізнавання, то дуже важливо вибрати такі ознаки та методи, які будуть давати найбільшу точність. Для кількісного розуміння ефективності розпізнавання використовуються стандартні метрики, наприклад, точність, recall, precision, F-метрика, ROC та інші.

У табл.2 наведені найбільш уживані методи для розпізнавання. Також наведені визначення деяких найбільш широко використовуваних ознак [16] сигналу Y = $\{y_1, ..., y_n\}$.

Табл. 2 Групи ознак, які можна виділити з даних натільних датчиків

Група	Метод
Часові ряди	Середнє, середнє відхилення, дисперсія, міжквартильний діапазон (IQR), середнє абсолютне відхилення (MAD), кореляція між осями, ентропія [18]
Частотні перетворення	Фур'є-трансформація (FT) [19] Дискретне косинусне перетворення (DCT) [20].
Інші	Основний компонентний аналіз (PCA) [21], лінійний дискримінантний аналіз (LDA) [22].

Вибір довжини вікна: поділ вимірюваного часового ряду на часові вікна є зручним рішенням для обробки числових рядів. При цьому ключовим фактором є вибір довжини вікна. Наявність досить коротких вікон може підвищити продуктивність системи, але в той самий час зменшить можливість розпізнавання складніших паттернів (короткі вікна можуть не надати достатньої інформації для повного опису виконуваної діяльності). І навпаки, якщо вікна занадто довгі, в одному часовому вікні може бути більше однієї активності [23]. У літературі використовуються різні довжини вікон: 0,08с, 1с, 1,5с, 3с, 5с, залежно від задачі що вирішується [24]. Звичайно, це рішення зумовлене типом

діяльності та вимірюваними ознаками. Наприклад, для сигналу частоти серцевих скорочень потрібні вікна 30 с [25]. Натомість для таких заходів, як ковтання, були застосовані вікна 1,5 секунди. Часові вікна також можуть або перекриватися [26] або неперервно йти один за одним [26]. Вікна, що перекриваються, призначені для більш точного керування переходами між інтервалами (Рис. 1.3).

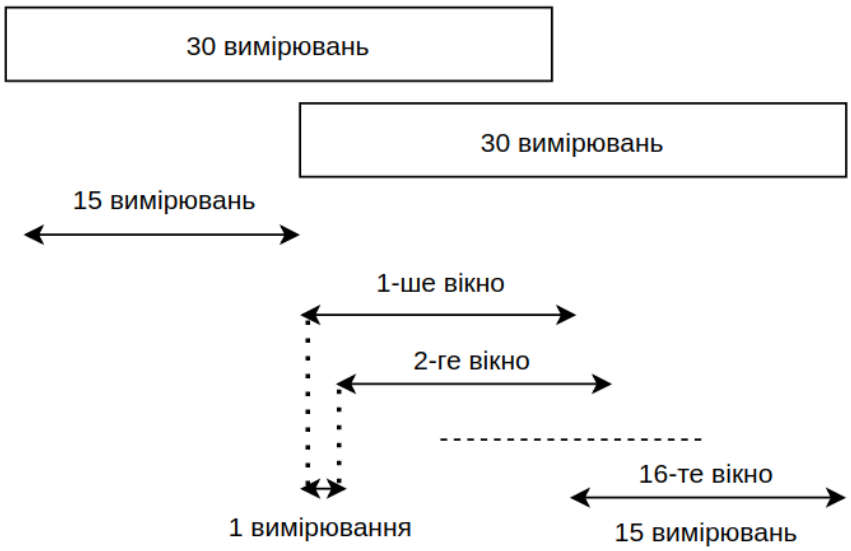


Рисунок 1.3. Часові вікна

Однак питання дослідження систем HAR за допомогою різних датчиків з подальшим їх впровадженням залишається відкритим та потребує додаткового опрацювання із систематизацією типів датчиків для HAR та методів збору даних.

1.6 Навчання моделей розпізнавання

Незважаючи на те, що останнім часом з розвитком технологій збирати дані стало значно простіше, основними викликами для систем HAR є розуміння контексту. Для цього використовують моделі машинного навчання, які інтерпретують зібрані дані і прогнозують діяльність.

Для навчання моделі використовують навчальний набір даних, який складається із зібраних даних з датчиків кількох людей, які містять опис необхідних типів активності. У випадку HAR кожен рядок в навчальному наборі

даних є вектором ознак, витягнутим із сигналів у часовому вікні. Приклади у навчальному наборі можуть бути або не бути позначені, тобто пов'язані з відомим класом (наприклад, ходьба, біг тощо). Існує два підходи до навчання — з вчителем та без вчителя, які стосуються відповідно розмічених та нерозмічених даних.

Оскільки система розпізнавання людської діяльності повинна класифікувати діяльність, то більшість моделей для систем HAR використовують навчання з вчителем.

Найпопулярніші алгоритми, які використовуються для класифікації типів активності:

- Дерева рішень: будують ієрархічну модель, в якій атрибути відображають на у вузлах, а ребра це можливі значення атрибутів (ребра є можливими значеннями атрибутів). Кожна гілка від кореня до вузла листя - це правило класифікації. Дерева рішень можуть бути оцінені в O (log n) для n атрибутів і зазвичай генерують моделі, які легко зрозуміти людині.
- Байєсівські методи: обчислюють ймовірності для кожного класу, використовуючи оцінені умовні ймовірності з навчального набору. Naïve Bayes (NB) [27] - основний представник цього сімейства класифікаторів. Ключовим питанням в Бейсівських мережах є побудова топології, оскільки необхідно зробити припущення про незалежність серед особливостей. Наприклад, класифікатор NB передбачає, що всі функції умовно незалежні з урахуванням значення класу, але таке припущення у багатьох випадках не відповідає дійсності. Наприклад, сигнали прискорення, фізіологічні сигнали сильно корелюються між собою.
- Метод опорних векторів (SVM) [28] також широко використовують в HAR, хоча вони не містять набору зрозумілих людині правил. SVM покладаються на функції ядра, які проектують всі екземпляри на більший розмірний простір з метою пошуку лінійної межі рішення для розподілу даних.

- Нейронні мережі та глибоке навчання: останнім часом дуже популярним вибором в якості моделі стають глибокі нейронні мережі, які мають різну архітектуру [29]. Так само як і SVN, вони не працюють за зрозумілими правилами, імітуючи роботу нейронів в мозку людини. До недоліків такого підходу треба віднести необхідність великого набору даних для початкового навчання. Оскільки нейронні мережі останнім часом отримали значне поширення, то можна розділити їх на різні типи, які відповідно використовуються для різних задач.

Нейронні мережі імітуючи роботу нейронів в людському мозку, складаються з базових елементів — перцептронів (Рис. 1.4). Самі мережі складаються з шарів перцептронів, які з'єднані між собою, і вхідними даними наступного шару є результати попереднього.

Перцептрон — математична модель, що перетворює вхідні сигнали за допомогою лінійного перетворення:

$$f(x) = sign(\sum_{i=1}^{n} w_i x_i - \theta)$$, [30]

де w —ваги вхідних сигналів, x — значення сигналів, θ – адаптивний поріг. Таким чином, перцетпрон видає значення "1", якщо лінійна форма вхідних сигналів та ваг більша, ніж θ, і "-1" в іншому випадку. Навчання перцептрона полягає в зміні вагових коефіцієнтів w, під час роботи з відомими даними.

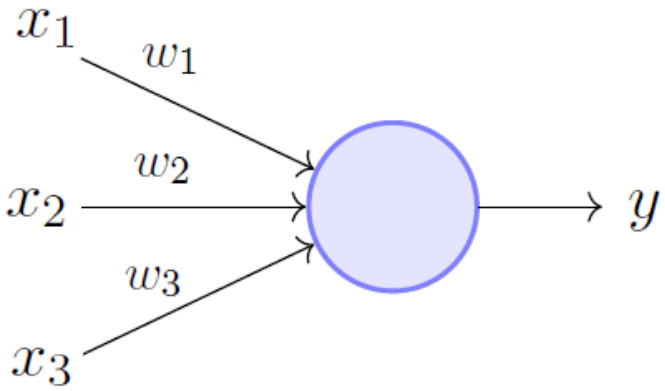


Рисунок 1.4. Перцептрон

Зазвичай для різних мереж використовуються різні функції активації. Тим не менше, необхідно слідкувати, щоб обрана функція мала наступні властивості [31]:

- нелінійність
- Неперервна диференційовність
- Монотонність.

Самими найбільш вживаними функціями є sigmoid, ReLU та softmax.

Sigmoid - це неперервно диференційована монотонна нелінійна S-подібна функція, яка визначається формулою:

$$f(x) = \frac{1}{1 + e^{-x}}$$

ReLU (rectified linear unit), або випрямляч є функцією активації, яка визначена таким чином (Рис. 1.5):

$$f(x) = \max(0, x)$$

Де x – вхідне значення нейрона.

Softmax - нормована експоненційна функція узагальнення логістичної функції, що "стискує" K-вимірний вектор Z із довільним значеннями компонент до K-вимірного вектора V(z) з дійсними значеннями компонент в області [0, 1] що в сумі дають одиницю. Функція задається наступним чином (Рис. 1.5):

$$V(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ для } j = 1, \ldots, K$$

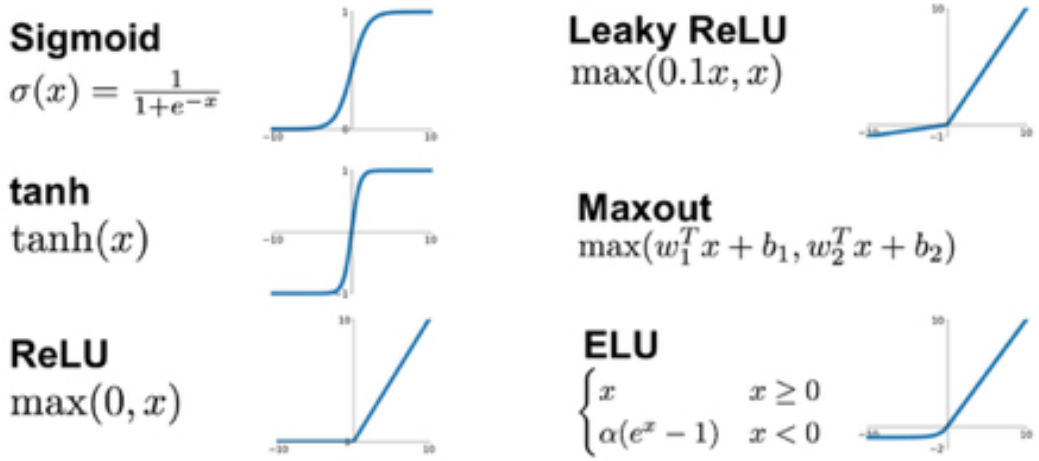


Рисунок 1.5. Функції активації

- Convolutional Neural Network (CNN): клас нейронних мереж, що містять згорткові шари, які містять нейрони, що виконують згортку на невеликих частинах вхідної інформації, отримуючи таким чином ознаки, що несуть інформацію про локальні структури [32] (Рис 1.6). Крім обробки зображень [33], розпізнавання звуку та обробки природних мов [34], CNN нещодавно почали застосовувати для обробки часових рядів у сенсорі HAR.

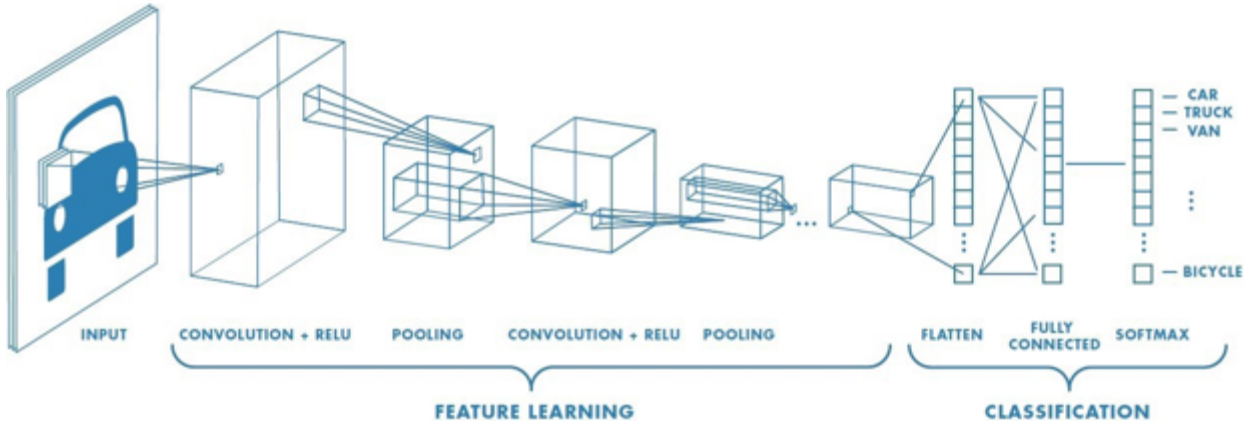


Рисунок 1.6. Приклад згорткової нейронної мережі (CNN) [8]

Рекурентні нейронні мережі (RNN) — нейронні мережі, що мають архітектуру, в якій з'єднання між вузлами утворюють граф, орієнтований в часі (Рис. 1.7 [35]). Таким чином у мережі з'являється "пам'ять". Ця властивість дозволяє таким мережам добре виконувати розпізнавання послідовностей, моделювати

довгострокові залежності у часових рядах, поширюючи інформацію через їх детермінований прихований стан.

Використання RNN дозволило отримати добрі результати у багатьох сферах, таких як послідовна класифікація зображень, аудіо класифікація [36], моделювання мови тощо [37].

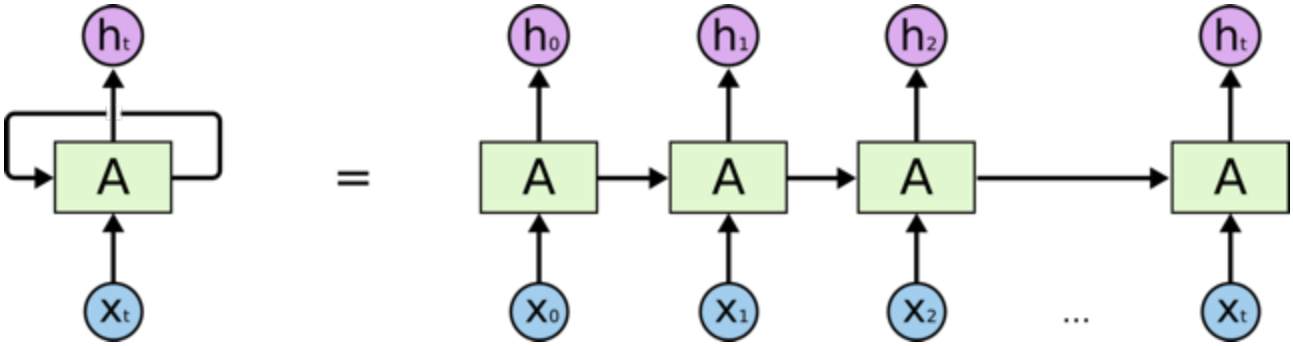


Рисунок 1.7. Приклад рекурентної нейронної мережі (RNN) [9]

- Long Short-Term Memory (LSTM) [38]: один з найуспішніших та найпоширеніших варіантів рекурентних нейронних мереж. Загальна архітектура LSTM складається з комірки (частина пам'яті блоку LSTM) та трьох "регуляторів", зазвичай званих воротами, потоку інформації всередині блоку LSTM: вхідний, вихідний і забуття. Зрозуміло, що комірка відповідає за облік залежностей між елементами вхідної послідовності. Вхідний потік керує тим, наскільки нове значення попадає в комірку, потік забуття керує тим, наскільки значення залишається в комірці, а вихідний потік керує тим, наскільки значення в комірці використовується для активації виводу обчислення (Рис. 1.8). Як функцію активації, часто використовують логістичну сигмоїдну функцію. Таким чином мережі LSTM широко використовуються для знаходження тимчасових залежностей у різноманітних задачах.

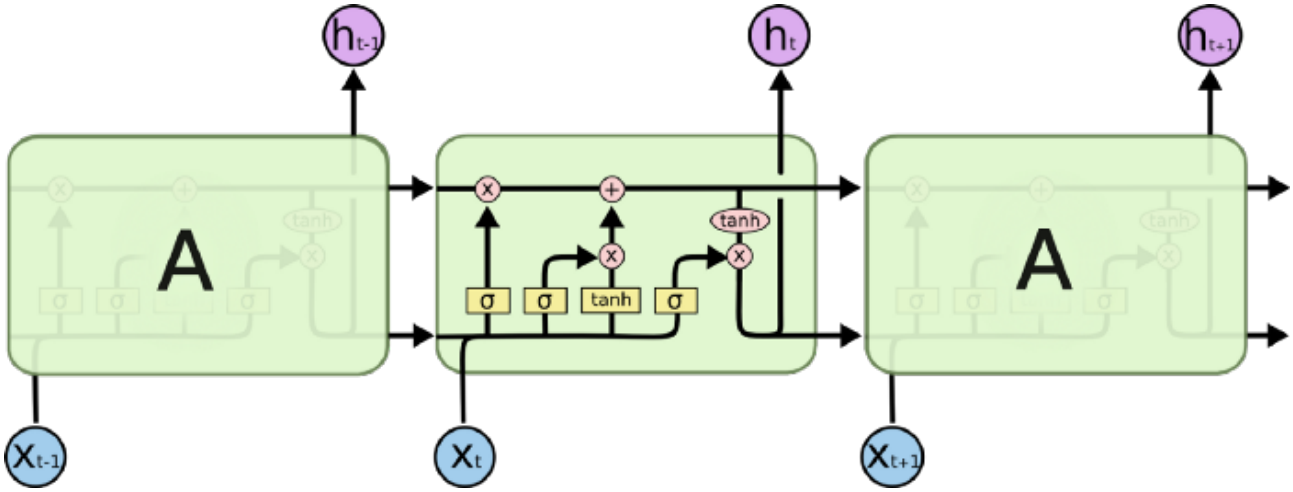


Рисунок 1.8. Приклад комірки LSTM [113]

Temporal Convolutional Networks (TCN) [39] — варіант згорткових нейронних мереж, який застосовується до довгих послідовностей (Рис. 1.9). Результати TCN завжди кращі, ніж LSTM / GRU для широкого спектру завдань. До переваг TCN можна віднести паралелізм, гнучкий розмір, відсутність проблеми вибухового градієнту, низькі потреби в пам'яті для тренувань, введення змінної довжини послідовностей. Архітектура може приймати послідовність будь-якої довжини і відображати її у вихідній послідовності тієї ж довжини, як і у рекурентних нейронних мережах [40].

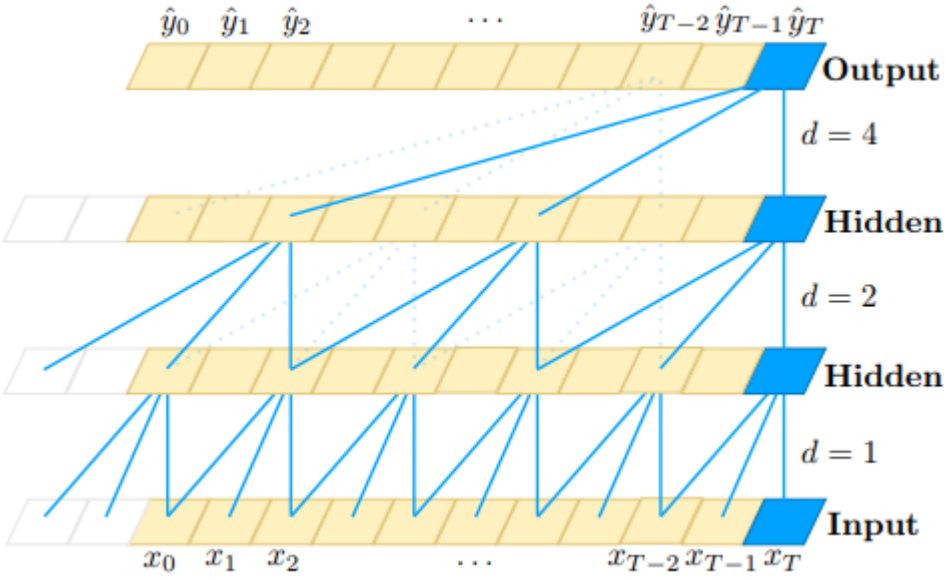


Рисунок 1.9. Приклад TCN [109]

Трансформери — архітектура нейронних мереж, що представлена в 2017 році, і швидко набирає популярність для певних задач [41] (Рис. 1.10). Зазвичай використовується в обробці природної мови. Як і рекурентні нейронні мережі (RNN), трансформери призначені для обробки послідовностей даних. Однак, на відміну від RNN, трансформери не вимагають, щоб послідовні дані оброблялися по порядку. Завдяки цій особливості трансформери дозволяють набагато більше паралелізації, ніж RNN, і тому скорочують час тренувань.

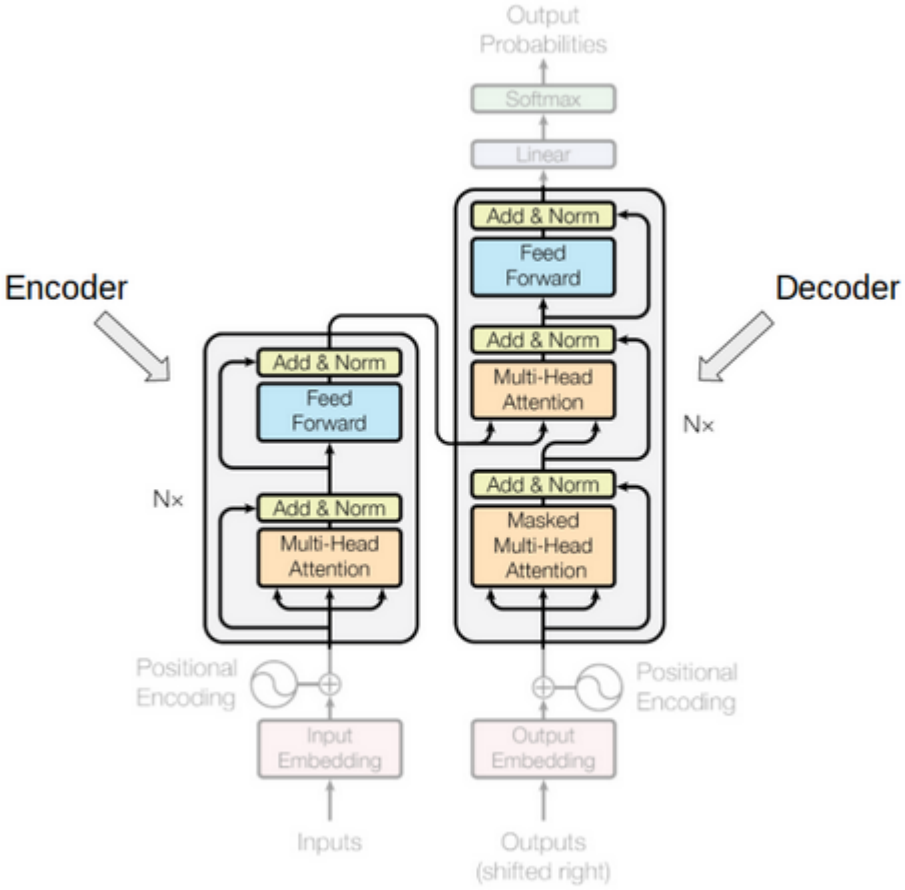


Рисунок 1.10. Приклад архітектури трансформерів [117]

Зазвичай, вибір однієї чи іншої моделі для класифікації залежить, як від набору даних, які використовується, так і від точності роботи цієї моделі. В більшості досліджень використовується крос валідація з статистичними тестами для валідації моделі [42]. При будь-якій класифікації можна виділити наступні категорії класифікованих даних:

- правильно позитивно класифіковані (True Positives, TP)
- правильно негативно класифіковані (True Negatives, TN)

- неправильно позитивно класифіковані (False Positives, FP)
- неправильно негативно класифіковані (False Negatives, FN)

Ці результати можна зібрати в матрицю, яка називається матрицею невпорядкованості. Це матриця розміру $n\ x\ n$, де n — кількість класів що класифікуються моделлю.

Сама проста метрика, яка використовується для валідації моделей для всіх класів — це точність (accuracy). Її можна визначити за наступною формулою:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Також, важливими показниками роботи моделі є метрика точності (precision) та повнота (recall). Точність показує відношення правильно класифікованих значень до всіх значень які були класифіковані позитивно до певного класу. Повнота в свою чергу показує відношення правильно класифікованих значень до всіх значень цього класу. Повноту та точність можна визначити за формулами:

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}$$

Дуже часто вимірюється F-метрика, яка поєднує повноту та точність в одному значенні:

$$F - measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Класифікація людської діяльності – це останній і найважливіший крок в HAR. Більшість людських рухів/жестів – це динамічні процеси. Один динамічний процес завжди складається з декількох кадрів. Щоб класифікувати динамічні процеси, класифікація людської діяльності повинна виконуватися після або разом з відстеженням людської діяльності.

Таким чином, [43] представляє таксономію HAR систем, які класифікуються в залежності від типу сенсорів і типу моделей, які використовуються (Рис. 1.11). Останнім поділом систем в запропонованій таксономії є поділ на системи реального часу та на аналіз типів активності не в реальному часі. Останнім часом, більшість систем HAR, які розробляються

різними виробниками та дослідниками, та які включаються в різні натільні пристрої (наприклад наручний годинник) є системами з розпізнаванням діяльності в реальному часі.

Рисунок 1.11. Таксономія HAR систем

Варто зазначити кілька систем, які, згідно з [43] досягли високої точності на даний момент:

- Мауер та ін. [44] представили систему розпізнавання активності eWatch , яка є пристроєм з мікроконтролером та вбудованими датчиками. Пристрій можна носити як спортивний годинник на зап’ясті. До комплекту входять чотири датчики: акселерометр, датчик світла, термометр та мікрофон. Використовуючи модель дерева рішення, загальна точність становила до 92,5% для шести різних типів активності. Такий формат пристрою досить популярний для занять спортом і багато компаній (наприклад Garmin, Polar, Suunto та інші) випускають свої продукти в форматі годинника для розпізнавання різних типів спортивної активності, хоча точність у таких годинників зазвичай менше.
- Вігіланте та інші [45] запропонували, мобільний додаток для розпізнавання людської діяльності в реальному часі на платформі

Android. Для вимірювання прискорення та фізіологічних сигналів, таких як частота серцевих скорочень, частота дихання, амплітуда форми дихання та температура шкіри, серед іншого застосовувався ремінець датчика BioHarness [46]. Як модель класифікатора використовувалася модель дерева рішень. Модель мала змогу визначати три дії із загальною точністю 92,6%, проте для цього людина не повинна виконувати ніякої іншої активності в той самий час.

- Рібоні та ін. [47] представили COSAR - основу для розпізнавання контекстної діяльності з використанням статистичних та онтологічних правил на платформі Android. Система розпізнає такі типи активності як: чищення зубів, ходьбу, стан спокою. COSAR збирає дані з двох акселерометрів, одного в телефоні та іншого на зап'ясті людини, а також з GPS мобільного телефону. Загальна точність була приблизно 93%
- Као та ін. [48] представили портативний пристрій для виявлення типу активності в реальному часі. Акселерометр розміщується на зап'ясті користувача. Як класифікатор застосовується лінійний дискримінантний аналіз (LDA) для зменшення розмірності простору функцій та відібрані вручну ознаки. Загальна точність становила 94,71% для таких типів активності: чистка зубів, удари, стук, біг, ходьба та розмахування.

Як можна побачити з опису цих систем, у них у всіх є спільний недолік — вони працюють тільки коли людина робить попередньо вивчені типи активності без одночасного використання інших типів активності. Наприклад, система яка розпізнає біг та чистку зубів, навряд чи буде спроможною розпізнати чистку зубів у людини, що скаче на коні. Більше того, деякі з описаних систем [48] показують високу точність тільки в штучно створеному середовищі, в той час як їх тестування за межами цього середовища показує точність не більшу за 70%. Таким чином, можна сказати, що основними викликами для HAR систем в сучасному стані є фільтрація шуму, яку створюють для сенсорів інші типи активності, а також покращення точності і зменшення неправильно позитивно класифікованих зразків.

Висновки до розділу 1

1. Формалізовано задачу визначення людської діяльності - діяльність людини в кожен момент може складатися з величезної кількості комбінацій різних рухів і тому виділити одну активність на фоні інших дуже складно. Визначення людської діяльності в першу чергу залежить від конкретної активності, яку необхідно визначити, але всі системи HAR в загальному мають схожу структуру, що дозволяє формалізувати задачу визначення людської діяльності для її подальшого аналізу, шляхом попередньої обробки отриманих даних та виділення корисних даних, які стосуються досліджуваної людської активності.
2. Здійснений аналіз існуючих моделей класифікації активності людини, серед яких більшість моделей для систем HAR використовують навчання. Серед відомих моделей одні - наглядні для людини, але з меншою точністю, інші - приймають умову, що всі функції, які стосуються людської активності, умовно незалежні, що в реальному житті неможливо. Найбільш популярним вибором моделі стали моделі на основі різних типів нейронних мереж, так як різні моделі на основі нейронних мереж дозволяють обробляти дані в реальному часі як послідовно, так і паралельно.
3. Здійснено порівняльний аналіз існуючих систем визначення людської діяльності, де відмічені як позитивні так і негативні сторони кожної з них. При цьому відмічено, що основним викликом сучасних систем HAR є фільтрація шуму, яку створюють для датчиків інші типи людської активності, а також покращення точності у визначенні конкретної людської активності.

Розділ 2. Методи отримання і оброблення сигналів сенсорів про дії людини

2.1. Порівняння сенсорів і їх можливостей

Згідно з законом Мура, кількість транзисторів на кристалі мікросхеми подвоюється кожні 24 місяці [49], таким чином за останні роки технології, що використовуються для виробництва різних сенсорів дуже розвинулися. Це також примінимо і для сенсорів, що використовуються в задачах HAR: таких як гіроскори, акселерометри, датчики тиску та інші. Розвиток нових технологій також дозволив цим сенсорам використовувати менше енергії, бездротову передачу даних, і менші батареї. Таким чином, велика кількість сенсорів почала застосовуватися для постійного моніторингу різних сфер діяльності людини — від моніторингу за трафіком до спостереженням за здоров'ям в реальному часі [50].

Таким чином, сучасні сенсори, які використовують і механічну, і електронну складову, називають MEMC - МікроЕлектроМеханічні системи [51]. Завдяки своєму розміру, вони застосовуються майже всюди, що дозволяє їх використовувати не тільки в лабораторіях, але й польових умовах [52]. Наприклад, крім MEMC, гіроскопи бувають механічні та оптичні. Але, незважаючи на розвиток технологій, вони складаються з великої кількості деталей, які в свою чергу мають малі допуски по розміру та складність збірки, що накладає певні обмеження на ціну та використання таких гіроскопів.

На відміну від механічних гіроскопів, які містять швидкообертове тверде тіло, яке має три обертальні ступені вільності, гіроскопи на основі технології MEMC використовують ефект Коріоліса для виміру зміни положення тіла. Зазвичай, в таких гіроскопах містяться 3 незалежних одновісних датчика вібрації, які реагують на обертання навколо відповідних осей X, Y, Z. Дві підвішені маси здійснюють коливання вздовж протилежних осей. З появою кутової швидкості, викликається зміна напрямку вібрації, яке фіксується ємнісним сенсором. Таким чином, сила Коріоліса, що діє на ці маси вираховується за формулою:

$F_c = -2m(w * v)$,

де m — маса, на яка здійснює коливання, w — кутова швидкість, v — швидкість маси.

В МЕМС гіроскопах сигнал підсилюється і фільтрується в результаті чого отримується напруга, пропорційна до кутової швидкості обертання. Такий гіроскоп може записувати до 8-10 тисяч вимірів за секунду. Кінцевий сигнал обробляється вбудованим 16-бітним процесором Для того щоб відокремити вібрації самого гіроскопу (вище 25 Гц), на отримані значення накладаються фільтри низьких частот [53].

Хоча гіроскопи на основі МЕМС і не такі точні, як оптичні, вони мають ряд незаперечних переваг [54]:

- малий розмір і вага (поміщається в смартфонах)
- низьке споживання енергії
- висока надійність
- дешевизна в виробництві
- відсутність необхідності в обслуговуванні
- майже відсутність вимог до середовища роботи

Якщо говорити про акселерометри, то акселерометри на основі МЕМС мають той самий принцип роботи, що й механічні та твердотільні аналоги. В основі будь-якого виду акселерометрів лежить другий закон Ньютона.

Існує 2 основних види МЕМС акселерометрів — механічні та струнні. Механічні — працюють за допомогою вимірювання зміщення маси. В такому випадку використовується різна маса для кожної з осей, яка зміщується при виникненні прискорення вздовж цієї осі (зміни фіксуються ємнісними датчиками). Наприклад, якщо такий акселерометр лежить на плоскій поверхності, то він покаже прискорення 0 по осям X та Y, і 1g прискорення по осі Z. Струнні акселерометри — вимірюють зміну частоти вібраційного елемента, спричинену зміною напруги.

У будь-яких акселерометрів є важливі характеристики:

- масштабний коефіцієнт (коефіцієнт пропорційності для лінійної залежності між вимірюваним прискоренням і вихідним сигналом, калібрується на виробництві)
- робочий діапазон частот
- порогова чутливість
- нелінійність (відхилення залежності між вихідним сигналом і прискоренням від лінійної при зміні прискорення)
- зсув нуля (різниця між показаннями приладу і проекцією гравітаційного прискорення на вісь при нульовому прискоренню, зазвичай вимірюється середньоквадратичне відхилення)

Всі переваги, які перераховані вище для МЕМС гіроскопа, також можна віднести і до акселерометра [55]. Однак, важливим моментом, який присутній у всіх МЕМС датчиках, є присутність шуму. Його можна класифікувати на кілька основних типів:

- зміщення
- шум внаслідок температурних впливів
- калібрування або нестабільність зсуву

Проведено багато досліджень для пошуку методів усунення похибок у гіроскопах та акселерометрах [56], [57].

Одним з найбільш впливових на шуми чинників є температура [58]. В роботі [58] проводиться аналіз впливу температури на характеристики МЕМС гіроскопа. Результатом дослідження є те, що нульова напруга гіроскопа містить короткочасний випадковий дрейф і довгостроковий дрейф. Для короткочасних шумів зазвичай використовуються деякі форми фільтра низьких частот або фільтра рухомого середнього для обробки високочастотних шумів у режимі реального часу. Результати показують, що з кожним підвищенням температури на 10 ◦C у гіроскопів спостерігається дрейф кутової швидкості приблизно 0,8 - 1,7 град / сек. Крім того, зазначено, що цей коефіцієнт не обов'язково однаковий для кожної осі. Оскільки зміна температури може впливати по різному на різні

прилади і на їх різні характеристики, то вплив таких ефектів можна побачити в характеристиках приладів. Враховуючи, що в дослідженні використовується MPU-9250, можна звернутися до його характеристик [8]:

- робоча температура приладу в діапазоні від -40◦C до 85◦C
- зміна коефіцієнта шкали чутливості від температури — ±0.04 %/◦C
- зміна чутливості лінійного прискорення до температури - ±0.01 %/◦C

Зважаючи на те, що ці значення вимірювалися для змін значень з температури 25◦C, то покази датчиків можуть відрізнятися з зміною температури.

Враховуючи, що прилад міститься близько до людського тіла, тобто при температурі близькій до 25◦C (при холодних температурах під верхнім одягом), то зміна температури зовнішнього середовища не має значних впливів на результат. Для того, щоб зовсім звести цей ефект до нуля, на великих інтервалах вимірювання, потрібно використовувати фільтр високих частот [59] .

Резистивний датчик механічної взаємодії (FRS) - це датчик, який дозволяє визначати фізичний тиск або вагу (Рис. 2.1.). Такий датчик являє собою пластину з тоістої полімерної плівки (PTF), що проявляє зменшення опору при збільшенні сили, прикладеної до активної поверхні [60]. Такі датчики не є точними, і не можуть використовуватися для отримання точних вимірювань [60].

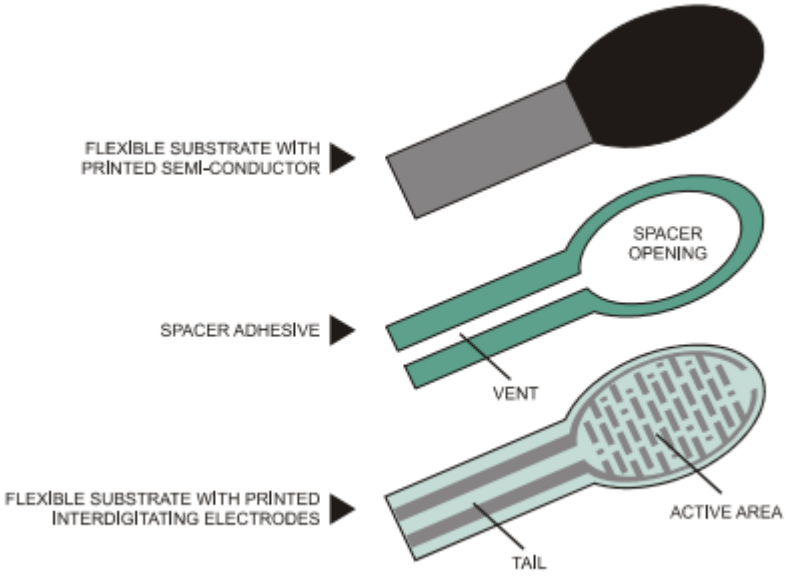


Рисунок 2.12. Будова FRS датчика [60]

Враховуючи, що виробник таких датчиків не надає гарантій точності, в експерименті такий датчик використовувався як бінарний індикатор руху грудної клітини для покращення точності моделі.

Розміщення сенсорів на людині може бути різним і відповідно для вимірювання різних рухів використовуються різне розміщення сенсорів. Наприклад, Тонг [61] в своєму дослідженні показав, що гіроскоп показує результати приблизно однієї точності, коли він розташований на одній площині відносно тіла людини. Загалом, необхідно враховувати розміщення сенсорів таким чином, щоб на них не впливали додаткові рухи, які можуть створювати шуми. Наприклад, якщо вимірюється ходьба, то самим логічним розміщенням сенсорів є нога, оскільки амплітуда потрібного сигналу найбільша, і в той самий час ноги найменше генерують додаткових сигналів, які можуть завадити розпізнаванню ходьби.

Якщо говорити про розміщення сенсорів гіроскопа та акселерометра на тілі людини для вимірювання частоти дихання — то найкращим місцем буде

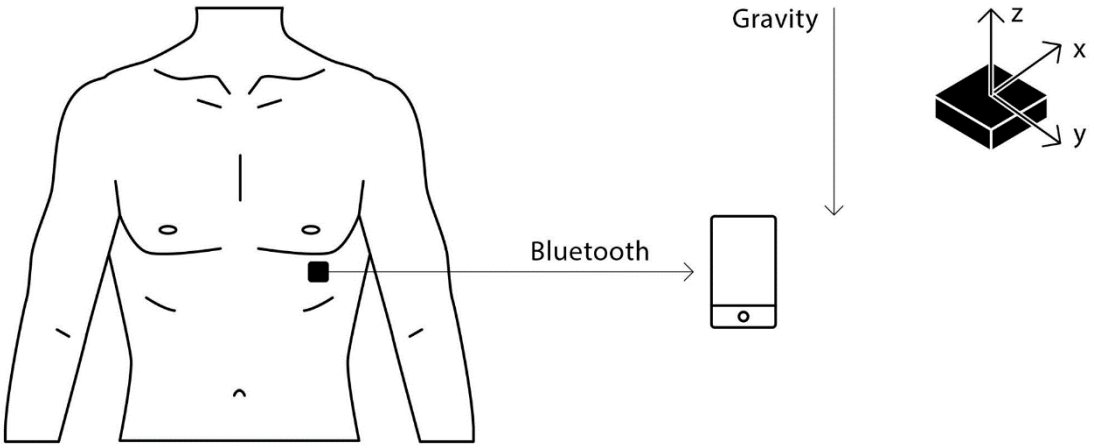


Рисунок 2.2. Місце кріплення сенсора для вимірювання частоти дихання [63]

частина грудної клітки, яка найбільше рухається під час дихання [62]. Такою частиною є розміщення на нижній частині грудної клітки збоку [63] (Рис. 2.2.). Після розміщення приладів з датчиками на тілі людини, наступає не менш важлива фаза — попередня обробка сигналів що збираються цих датчиків в реальному часі. Це один із самих важливих етапів роботи з аналізу діяльності людини в реальному часі. Оскільки людина постійно робить додаткові рухи, крім тих, які необхідно виміряти, то зібрані сигнали необхідно якісно фільтрувати для того щоб виключити вплив інших рухів на вимірювану активність.

У випадку вимірювання частоти дихання, всі рухи людини є додатковими, і такими що генерують додатковий шум для вимірювання сигналу дихання. Таким чином попередня обробка сигналів у випадку вимірювання частоти дихання набуває особливої ваги.

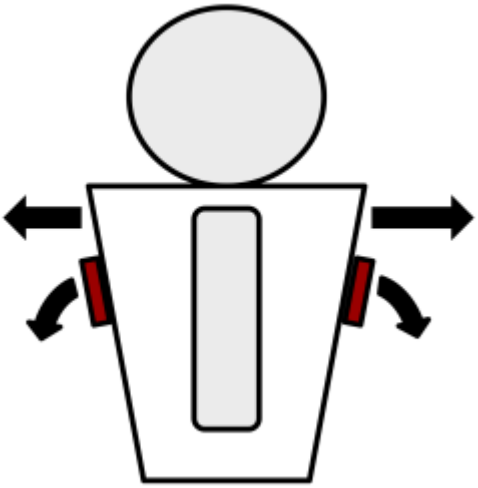


Рисунок 2.3. Система з двох акселерометрів що використовується для вимірювання частоти дихання [23]

Для того щоб провести таку попередню обробку сигналів, використовуються різні методи. Наприклад для виділення сигналу дихання Вертенс та інші [64] пропонують використати систему з двох акселерометрів (Рисунок 2.3). Робота цієї системи базується на рухах грудної клітини, а також тому, що спина при цьому залишається нерухомою. Таким чином, в цій системі один з акселерометрів використовується для виміру руху грудної клітини людини, а інший — для компенсації всіх інших рухів, що робить людина під час вимірювання. В результаті цього, прискорення грудної клітини людини визначається як різниця прискорень двох акселерометрів:

$a = a_1 — a_2$, де a_1 — акселерометр що прикріплений спереду, а a_2 — на спині.

Після фільтрування отриманого сигналу, в [64] пропонують знаходити піки кожної амплітуди, рахувати час між цими піками і таким чином вирахувати частоту дихання.

Такий метод має як плюси, так і мінуси. До плюсів можна віднести можливість досить точно відфільтровувати додаткові рухи людини (в експерименті вдалося досягти точності визначення частоти дихання на рівні 90% навіть при бігові), досить простий метод визначення що не має складних обчислювальних алгоритмів, за рахунок він споживає досить мало енергії і може бути використаний на вбудованих платформах.

Так само в даного метода є й мінуси, такі як:

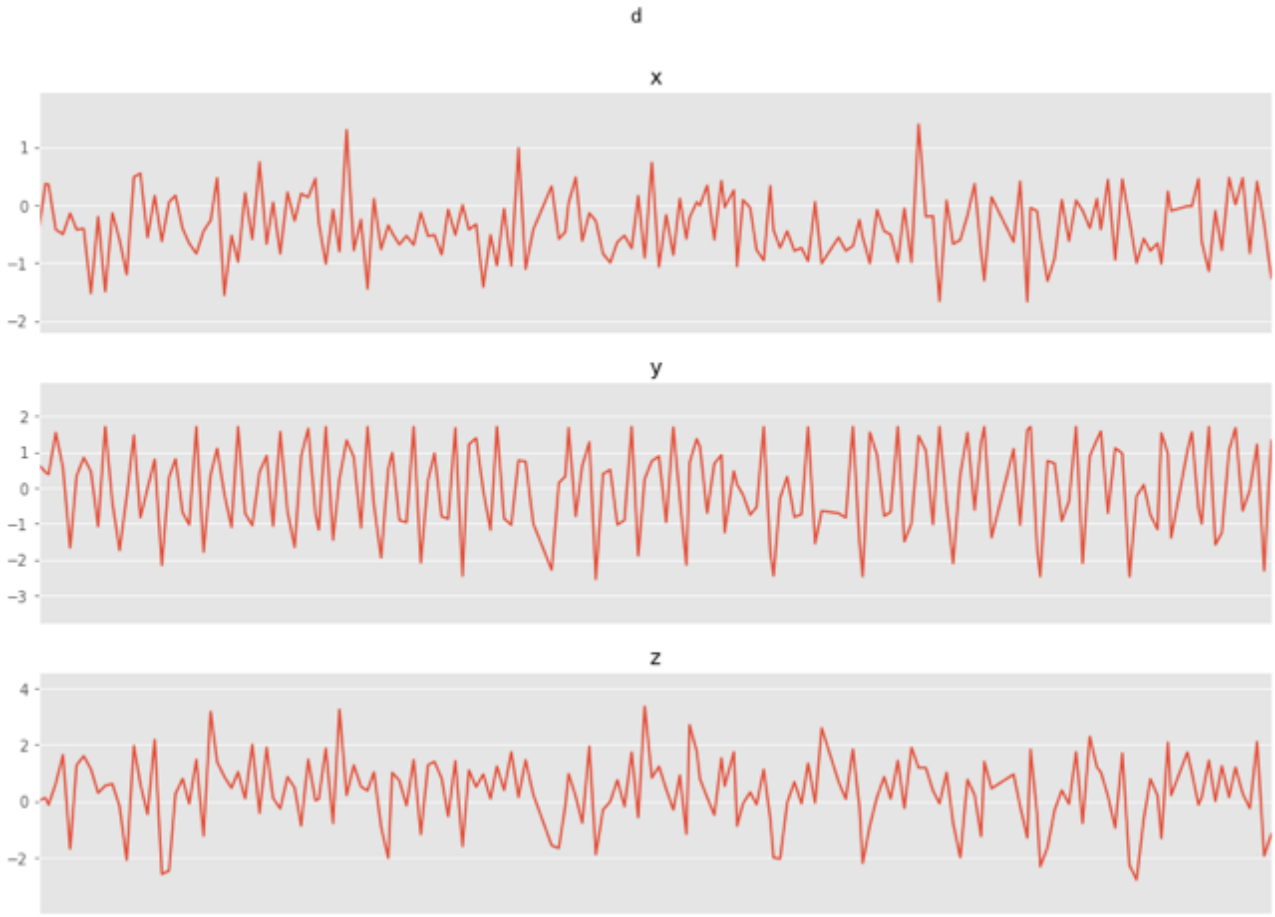


Рисунок 2.4. Приклад необробленого сигналу з акселерометра

- використання одразу двох датчиків, які мають бути закріпленими на людині
- обробка сигналів від двох окремих датчиків має бути неперервним для функціювання системи

Для того щоб уникнути цих мінусів, в [65] пропонують використовувати датчик деформації, в [25] пропонують використати тільки один трьох-вісний акселерометр.

Якою б не була система датчиків, необроблений сигнал акселерометра має вигляд, як показано на Рисунку 2.4

Як видно з Рисунку 2.4, такий сигнал необхідно обробити перед тим як виділяти з нього частоту дихання. Для обробки сирих даних з сенсорів використовуються такі методи як: вейвлет-перетворення, накладання фільтру, середнє значення, дискретне перетворення Фур'є, передавальна функція та інші.

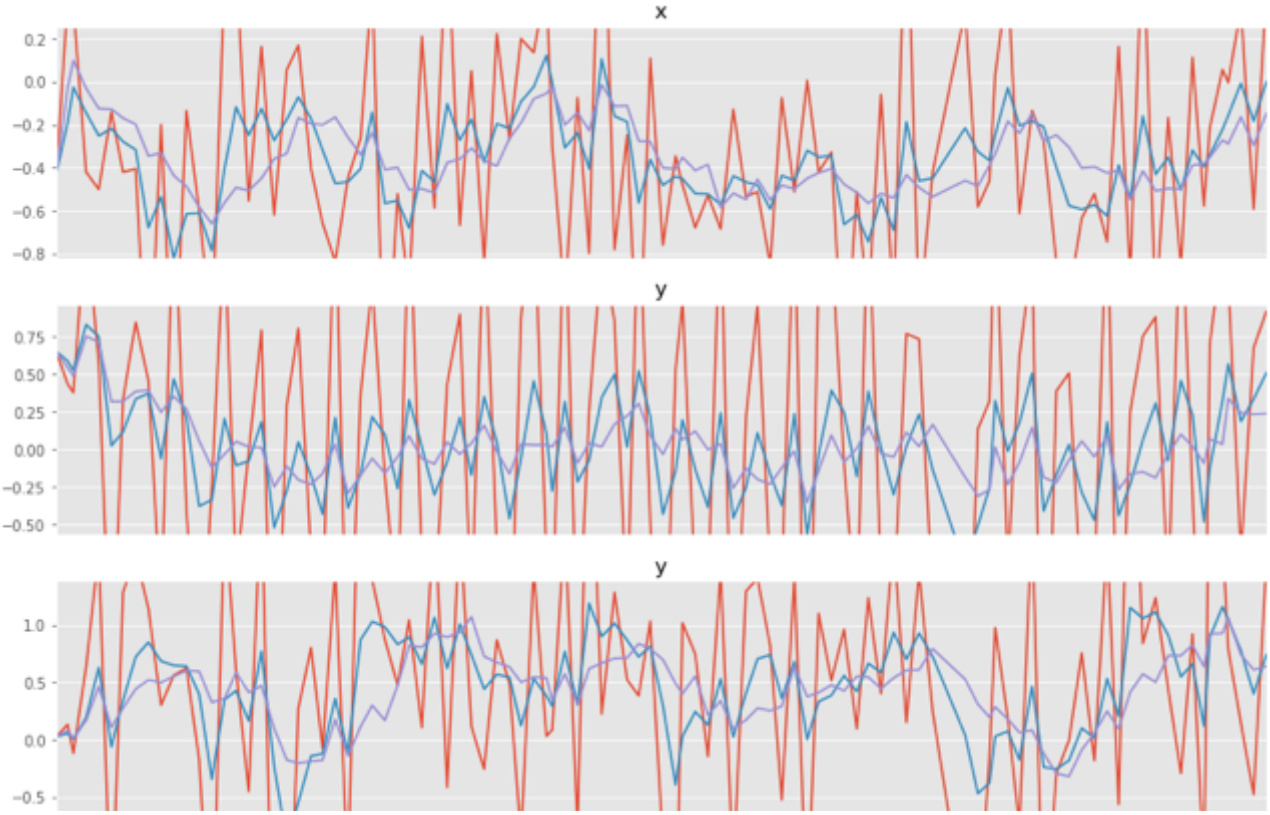


Рисунок 2.5 Приклад обробки сигналу використовуючи рухоме середнє та експотенційне рухоме середнє

При використанні будь-якого з цих методів необхідно враховувати доцільність його використання, а також його вплив на необхідну компоненту вхідного сигналу.

2.2. Рухоме середнє

Рухоме середнє — метод що використовується в аналізі часових рядів для згладжування коротких коливань і виділення інших компонентів (тренду, сезонів, циклів, тощо) що присутні в даних. Рухоме середнє рахується шляхом усереднення даних часових рядів протягом k періодів часу (Рис. 2.5.). Вибір розміру вікна M залежить від бажаного рівня згладжування, оскільки при збільшенні значення M покращує згладжування за рахунок точності:

$$SMA_t = \frac{x_t + x_{t-1} + x_{t-2} + \cdots + x_{M-(t-1)}}{\mathrm{M}}$$

2.3. Експотенційне Рухоме середнє

Експотенційне Рухоме середнє — метод, що часто використовується для фільтрації шумів та виявлення трендів. В цьому методі вага кожного елемента з

часом поступово зменшується. Тобто більшої ваги набувають останні дані. Порівняно з простим рухомим середнім, цей метод більш чутливий до змін:

$$EMA_t = \begin{cases} x_0 & t = 0 \\ \alpha x_t + (1-\alpha) EMA_{t-1} & t > 0 \end{cases}$$

де $\mathbf{x_t}$ **-** спостереження в час t, $\mathbf{EMA_t}$ **-** експоненціальна рухома середня за період часу t, **α** - згладжуючий коефіцієнт, що має значення від 0 до 1 і являє собою вагу, застосовану до останнього періоду**.**

Як видно з Рисунку 2.5, використання рухомого середнього значно зменшує амплітуду сигналу, а також прибирає всі пікові значення даних, тому не може використовуватися для визначення частоти дихання.

2.4. Перетворення Фур'є

Перетворення Фур'є розкладає сигнал в частоти використовуючи ряд сисусних хвиль [66] (Рис. 2.6.). Це допомагає переходу між часовою та частотною областями. Тим не менше, таке перетворення має ряд обмежень для застосування з нестаціонарними сигналами.

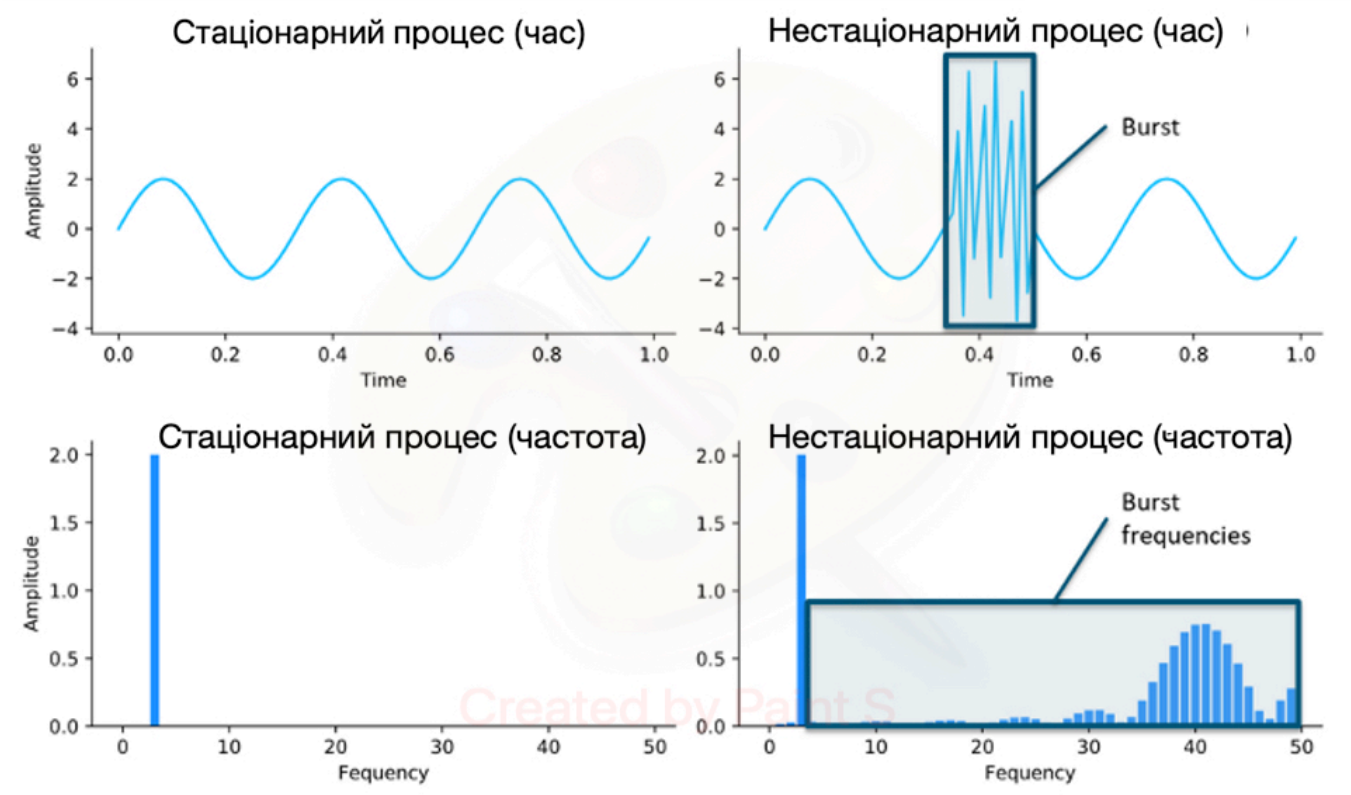


Рисунок 2.6 Перетворення Фур'є для стаціонарного та нестаціонарного процесів

Для того щоб продемонструвати ці обмеження, можна привести приклад перетворення Фур’є для стаціонарного сигналу, та нестаціонарного (Рис). Так як стаціонарний сигнал не міняє свого середнього значення, дисперсії та коваріації з часом, то, як можна бачити з Рисунку , перетворення Фур’є показує амплітуду сигналу як функцію часу, яка може бути представлена як окремі ізольовані частоти в частотній області. У випадку ж з нестаціонарним сигналом, неможливо визначити відповідність частоти сигналу до часової області, оскільки перетворення Фур’є використовує тільки частотну характеристику сигналу.

Що в свою чергу означає, що неможливо отримати повну інформацію про нестаціонарний сигнал в часовій області використовуючи тільки перетворення Фур’є.

2.5. Смугові фільтри

Фільтр низьких частот, передає сигнали з частотою, нижчою за певну частоту відсічення, і послаблює сигнали з частотами, вищими за граничну частоту [67]. Відповідно фільтр високих частот, передає сигнали з частотою, що перевищує частоту відсічення, і послаблює сигнали з частотами, меншими за частоту відсічення. Смуговий фільтр може бути сформований каскадним фільтром високих і низьких частот. Смуговий фільтр відхилення - це паралельна комбінація фільтрів низьких і високих частот [67].

2.5.1. Фільтр Баттерворта (Butterworth Filter)

Фільтр з максимально плоскою (без пульсацій) АЧХ в області пропускання і наближається до 0 в зоні зупинки [68]. Завдяки такій характеристиці АЧХ, такий фільтр є одим з найпопулярніших фільтрів низьких частот.

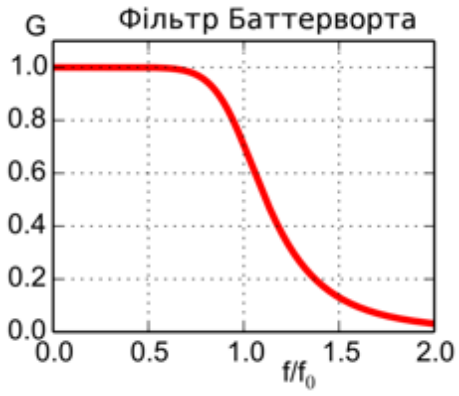


Рисунок 2.7 Фільтр Баттерворта

Враховуючи експеримент, проведений [64], з двома акселерометрами, було вирішено використовувати запропонований в [64] підхід до фільтрації сигналу. Для зменшення шумів від сенсорів, був використаний фільтр Баттерворта (Рис. 2.7.). Враховуючи визначену в [69] частоту прискорення дихання, були виставлені частоти фільтрації:

$$F_{low} = 0.1 \text{ Hz}$$

$$F_{high} = 0.8 \text{ Hz}$$

Тим не менше, після використання цього фільтру, все ще залишається забагато шуму. Тому було прийнято рішення використати адаптивний фільтр запропонований [64] з наступним алгоритмом:

1. Використання фільтра Баттерворта (з смугою пропускання 0.1-0.8 Гц)
2. Проводиться спектральна характеристика відфільтрованого сигналу за допомогою швидкого перетворення Фур'є
3. Отримана максимальна частота з перетворення Фур'є використовується для побудови нового 4-х ступеневого рекурсивного фільтру з полосою пропускання:

$$f_l = f_{max} - f_{bw}$$

$$f_h = f_{max} + f_{bw}$$

де f_{bw} – змінний коефіцієнт, що вибирається в залежності від виду діяльності і дисперсії дихання ([70] пропонує використовувати 0.25 для малорухомих дій і 0,50 Гц для більш рухомих)

2.6. Peak Detection

При фільтрації сигналів знятих з сенсорів, виникає потреба виявлення і обробки аномалій. Для цього використовуються алгоритм розпізнавання піків в сигналі, а також моделі машинного навчання.

Алгоритм розпізнавання піків в часових рядах статистичний [71], і базується на дисперсії – якщо нова точка ряду знаходиться більше ніж на стандартне відхилення від рухомого середнього значення, то алгоритм сигналізує про знайдений пік. За рахунок того що даний алгоритм використовує рухоме середнє та відхилення окремо створене для кожного вікна, то незважаючи на рівень сигналу, знайдені піки мають приблизно однакову точність.

Алгоритм приймає на вхід 3 значення: lag = відставання рухомого вікна, поріг спрацьовування та коефіцієнт впливу. Алгоритм можна записати за допомогою наступного псевдокоду [71]:

```
# Settings (the ones below are examples: choose what is best for your data)
set lag to 5;          # lag 5 for the smoothing functions
set threshold to 3.5;  # 3.5 standard deviations for signal
set influence to 0.5;  # between 0 and 1, where 1 is normal influence, 0.5 is half
# Initialize variables
set signals to vector 0,...,0 of length of y;   # Initialize signal results
set filteredY to y(1),...,y(lag)              # Initialize filtered series
set avgFilter to null;                        # Initialize average filter
set stdFilter to null;                        # Initialize std. filter
set avgFilter(lag) to mean(y(1),...,y(lag));    # Initialize first value
set stdFilter(lag) to std(y(1),...,y(lag));     # Initialize first value
for i=lag+1,...,t do
  if absolute(y(i) - avgFilter(i-1)) > threshold*stdFilter(i-1) then
    if y(i) > avgFilter(i-1) then
      set signals(i) to +1;                   # Positive signal
    else
      set signals(i) to -1;                   # Negative signal
```

```
    end
    set filteredY(i) to influence*y(i) + (1-influence)*filteredY(i-1);
  else
    set signals(i) to 0;                    # No signal
    set filteredY(i) to y(i);
  end
  set avgFilter(i) to mean(filteredY(i-lag),...,filteredY(i));
  set stdFilter(i) to std(filteredY(i-lag),...,filteredY(i));
end
```

lag: параметр lag визначає, наскільки дані будуть згладжені та наскільки адаптивний алгоритм до змін. Для стаціонарних сигналів **lag** має бути якомога більшим. Для нестаціонарних сигналів необхідно адаптувати параметр до частоти зміни сигналу (в ході експерименту було визначено що для визначення піків в сигналі дихання найкраще використовувати значення 90, враховуючи що акселерометр має частоту 100 вимірів в секунду)

коефіцієнт впливу (influence): параметр, який визначає вплив нових значень на поріг спрацьовування. Значення цього параметру встановлюється від 0 до 1. Якщо значення 0, то сигнал стаціонарний, і рухоме середнє та стандартне відхилення не міняється. Якщо сигнал не стаціонарний – то необхідно вибирати середнє значення в залежності від природи сигналу (якщо сигнал призводить до структурного розриву [72], то значення має бути близьке до 1). Для сигнала з сенсорів використовується 0.5 [73]

поріг спрацьовування (**threshold)**: це кількість стандартних відхилень від рухомого середнього, вище якого алгоритм класифікує нову точку даних як пік. (параметр встановлюється з очікування кількості піків за проміжок часу і відповідає за чутливість алгоритму).

2.7. Автоенкодери

Іншим методом виявленя аномалій є використання моделі автоенкодера [74].

Автоенкодер - це особливий тип нейронної мережі, який копіює вхідні значення до вихідних значень, як показано на Рис. Фактично, автоенкоер являє собою алгоритм стискання з певними властивостями:

- Він залежний від даних (це означає, що він може стискати лише дані, схожі до навчального набору)
- Має втрати (це означає, що декомпресовані дані будуть мати похибку в порівнянні з вхідними)
- Навчається без вчителя на попередніх прикладах

Кількість нейронів у прихованому шарі менша ніж у вхідному. Це пояснюється тим, що якщо кількість нейронів у прихованих шарах менше, ніж у вхідних шарах, приховані шари запам’ятовують тільки важливу інформацію з вхідних значень [74] (Рис. 2.8.). Це змушує приховані шари вивчати наявні шаблони з вхідних даних та ігнорувати “шуми”.

Якщо кількість нейронів у прихованих шарах перевищує кількість нейронів у вхідних шарах, нейронній мережі буде надано занадто багато можливостей для вивчення даних. В такому випадку, автоенкодер просто скопіює всі вхідні дані.

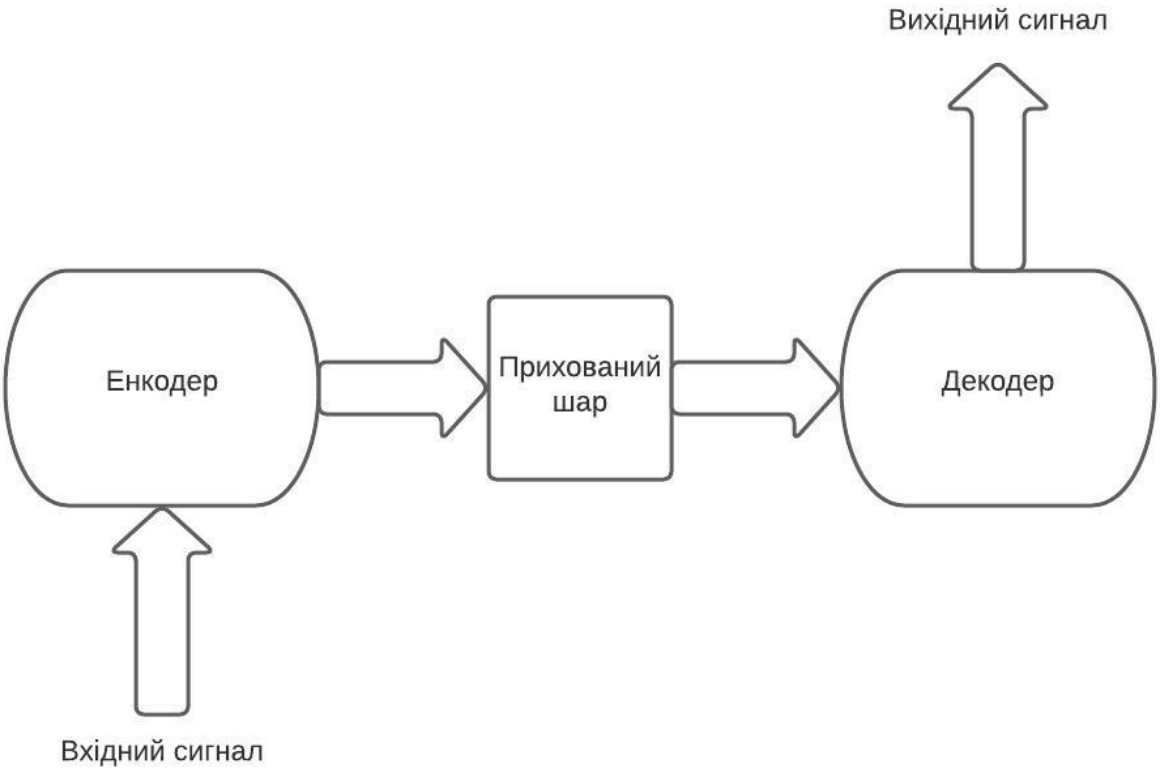


Рисунок 2.8 Архітектура автоенкодера

Для виявлення аномалій за допомогою автоенкодера використовують такий алгоритм [75]:

- Визначення похибки на тренувальному наборі даних
- Вибір значення максимальної похибки як порогового значення
- Якщо вхідне і вихідне значення точки відрізняється більше ніж на порогове значення, точка позначається як аномалія [76]

Автоенкодери застосовуються не тільки для пошуку аномалій, а й для зменшення розмірності. Джеффрі Хінтон [77] показав, що навчений автоенкодер дає меншу похибку порівняно з першими 30 основними компонентами PCA та краще розділення кластерів.

Автоенкодери також мають широке застосування в комп'ютерному зорі та редагуванні зображень. Вони використовуються при розфарбовуванні зображень для перетворення чорно-білого зображення в кольорове, для усунення шумів [78].

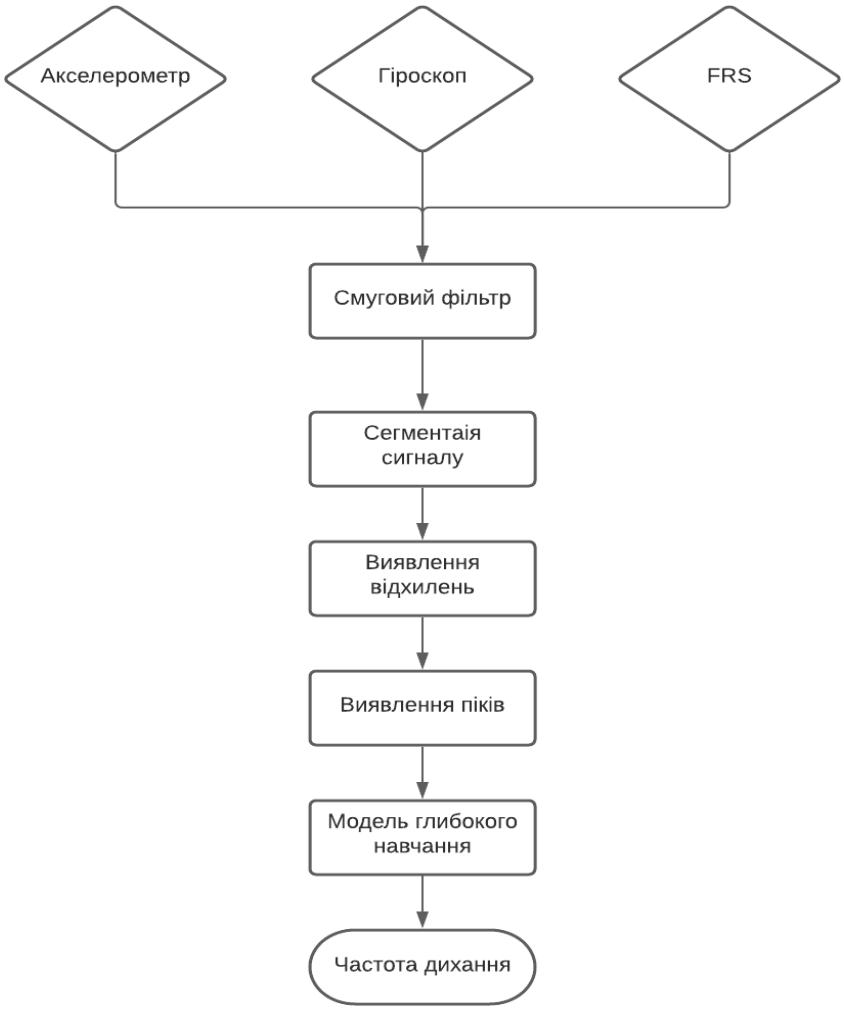


Рисунок 2.9. Алгоритм обробки сигналу з сенсора

Це досягається за рахунок того, що PCA використовує лінійні перетворення. В той час, як автоенкодери можуть виконувати нелінійні перетворення за допомогою нелінійної функції активації та кількох шарів [79].

Враховуючи всі наведені методи фільтрації сигналу, не вдалося виділити один з них, який би точно визначав рух що характеризує дихання. Також, слід зазначити що для різних сенсорів було застосовано різні алгоритми фільтрації, враховуючи різну природу сигналів з цих сенсорів. Однак застосування ансамблів з наведених в даному розділів методів дозволило виділити елементи руху дихання з сигналів сенсорів.

Зокрема, для акселерометра використовувалися фільтр Баттерворта з частотами фільтрації, запропонованими в [64], а також виявлення аномалій за допомогою швидкого перетворення Фур’є та моделі енкодера-декодера.

Для гіроскопа використовувалася комбінація з адаптивного рухомого середнього та смугового фільтру.

Для FRS сенсора був застосований алгоритм пошуку піків, оскільки зважаючи на похибку сенсора використовувати значення його сигналу не є доцільним, в той час як наявність піків показує зміну стану, що використовується в якості додаткової ознаки в моделі визначення типу дихання.

Таким чином можна скласти алгоритм обробки та класифікації сигналу з сенсорів (Рис. 2.9.).

Звичайно, навіть виділивши ознаки руху з сигналу сенсорів, не можна однозначно передбачити коли людина рухається, а коли просто дихає. В випадку проведеного експерименту для такого визначення було створено набір даних, що складався з записів акселерометра, гіроскопа та FRS сенсора в різних станах людини.

Висновки до розділу 2

1. Розглянуто сенсори отримання даних про рухи людини – сенсори розділено на групи по методу збору інформації та проаналізовано їх здатність до збору даних про рух людини. В першу чергу розглянуто натільні сенсори, їх параметри, можливості. Використання того чи іншого сенсора та його місце розташування в першу чергу залежить від руху, який необхідно визначити. Хоча рухи людини й мають схожу структуру, однак потребують окремих підходів до збору та обробки даних.
2. Здійснено аналіз методів збору даних з сенсорів та методів їх попередньої обробки. Так як рух людини складається одночасно з кількох рухів, а також з взаємодії з навколишнім середовищем, то дані отримані з сенсорів необхідно обробити перед класифікацією. Розглянуто різні методи попередньої обробки даних з сенсорів, зокрема: різні вид рухомого середнього, перетворення Фур’є, смугові фільтри та фільтр Баттерворта. Зроблено порівняння цих методів обробки даних для задачі визначення типу дихання людини.

3. Однією з важливих задач при обробці даних з сенсорів є визначення аномалій. Аномалії можуть свідчити або про різку зміну положення людини (наприклад при падінні значення прискорення значно збільшується), або про похибку вимірювання. Для задачі визначення типу дихання людини, важлива зміна значень даних з сенсора в продовж певного часу, тому виникає задача виявлення аномалій. Розглянуто та зроблено порівняння статистичного методу пошуку аномалій в сигналі та автоенкодера.
4. Як результат проведеного аналізу запропоновано алгоритм попередньої обробки вхідних даних з сенсорів для подальшого використання в класифікації дихання

Розділ 3. Порівняння алгоритмів машинного навчання

3.1. Класифікація методів машинного навчання.

Методи машинного навчання можна розділити за різними ознаками. Частіше за все, їх класифікують на статистичні методи машинного навчання та глибоке навчання (deep learning) [80].

Рисунок 13 Таксономія методів машинного навчання

Також, для статистиних методів, є популярною класифікація на методи з вчителем та без, а також гібридні (upervised, unsupervised та semi-supervised) (Рис. 3.1), але з зростанням кількості систем з постійним навчанням (online learning), така класифікація стає менш популярною [81]. При цьому слід враховувати, що системи які використовують машинне навчання дійшли до того рівня що рідко використовують лише одну модель яка складається з одного методу. Значно частіше — це пейплайн, що включає в себе кілька моделей і трансформацій даних до того як видати кінцевий результат. Також популярними залишаються ансамблі моделей, які включають в себе кілька моделей підряд, що використовують результати роботи однієї моделі як вхідні дані для іншої.
Незважаючи на це, як пейплайни, так і ансамблі моделей, атомарно складаються з простих моделей та методів. Це можуть бути комбіновані статистичні методи разом з методами глибокого навчання.

Розглянемо детальніше навчання з вчителем (Рис. 3.2):

Навчанням з вчителем називають навчання, де є вхідні змінні (X), та цільова змінна (Y) і алгоритм спроможний апроксимувати таку функцію *f(X)*, що:

$Y = f(X)$ [82]

Таким чином, коли з'являються нові вхідні дані, така модель має змогу передбачити значення цільової змінної Y для нових даних з певною точністю.

Такий процес називається навчанням з вчителем, оскільки навчання алгоритму на основі початкового навчального набору (X) даних можна сприймати як вчителя, який контролює процес навчання. Оскільки наперед відомі правильні відповіді, алгоритм ітеративно робить передбачення щодо навчальних даних і

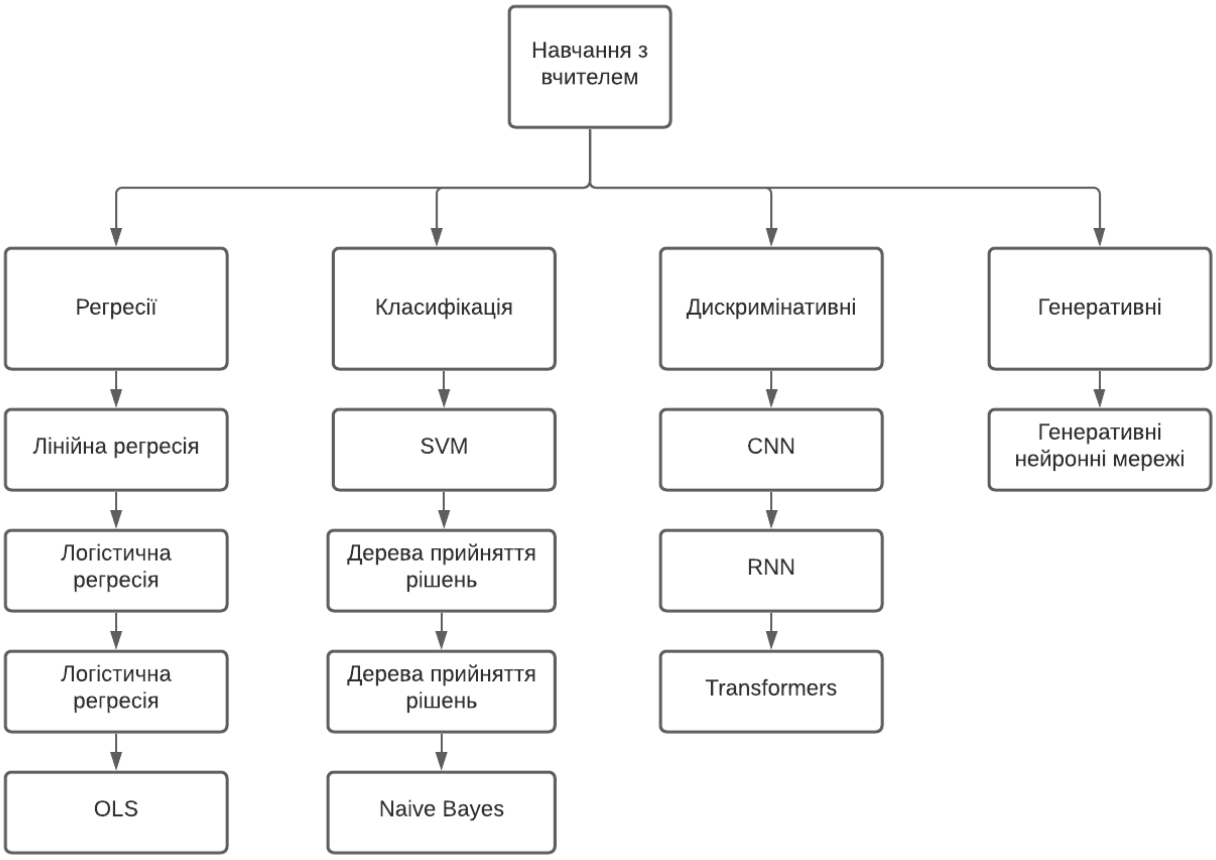


Рисунок 14 Таксономія методів машинного навчання з вчителем

відповідно коригується у відношенні до початкового набору даних. Навчання припиняється, коли алгоритм досягає визначеного рівня точності передбачення.

Навчання без вчителя (Рис. 3.3):

Відрізняється від навчання з вчителем тим, що при наявності вхідного набору даних (X), немає цільової змінної Y. Основні задачі, які вирішує навчання без вчителя є [83]:

- кластеризація — групування даних навколо певних признаків, які алгоритм визначає за допомогою використання розподілу даних (наприклад k-nn, k-means)
- Асоціація — створення правил, які б описували більшу частину даних (apriori) [84]

Навчання без вчителя рідко використовується як єдина модель в пейплайні. Часто дані кластеризуються для того щоб використати їх приналежність до того чи іншого кластера у вигляді додаткового параметру для наступної моделі.
Навчання частково з вчителем (Рис. 3.4) – рідко використовується, в основному в якості додаткового навчання.

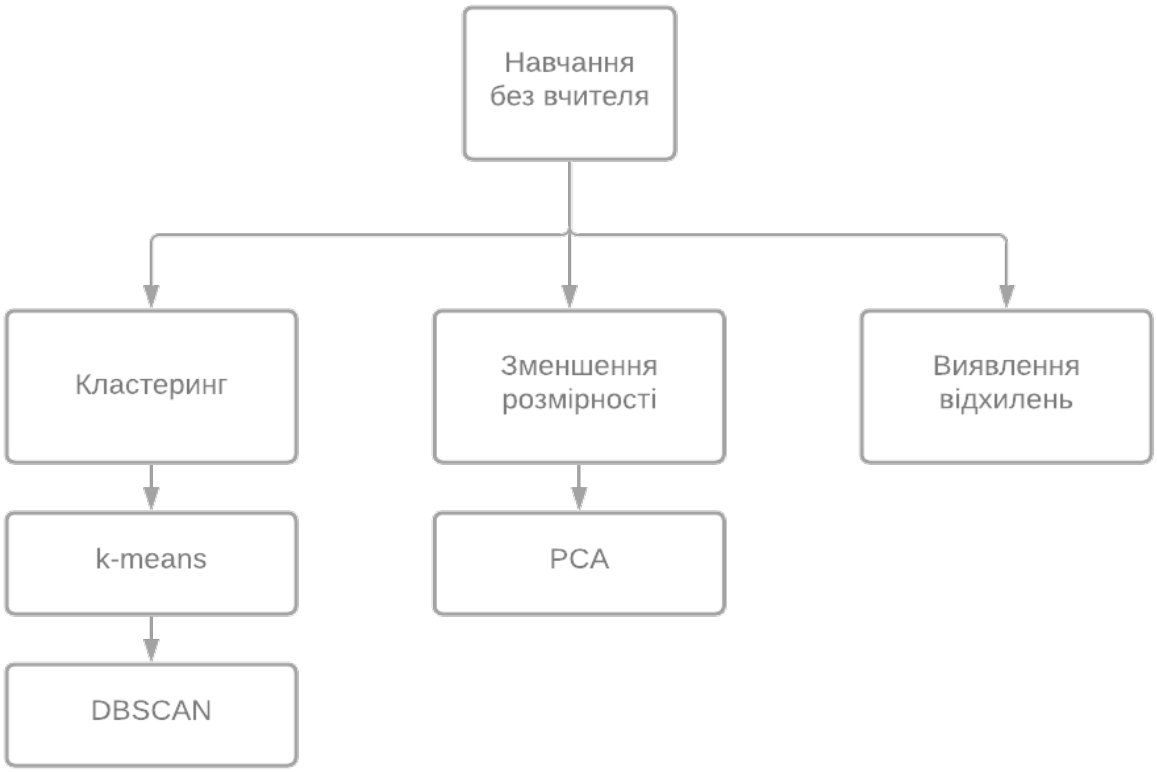


Рисунок 15 Таксономія методів машинного навчання без вчителя

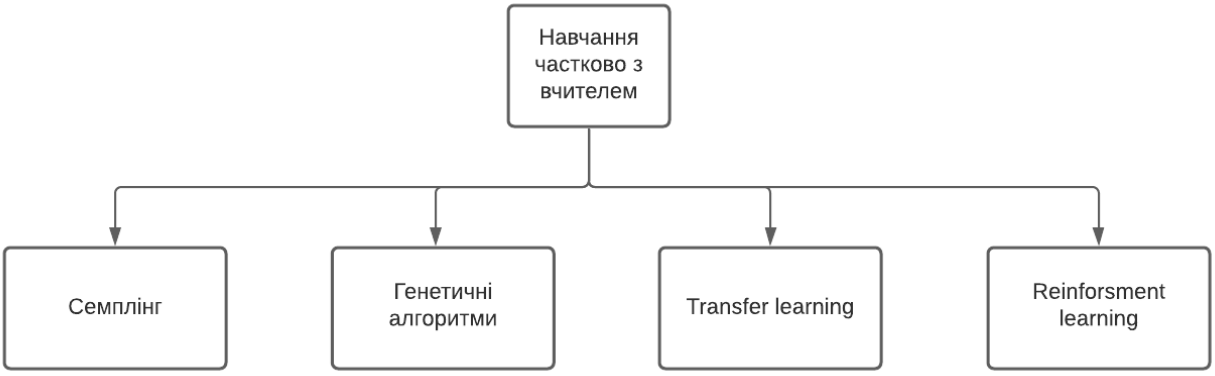


Рисунок 16 Таксономія методів машинного навчання частково з вчителем

3.2. Статистичні методи класифікації часових рядів

Розглядаючи задачу визначення людських дій (HAR), можна сказати що в частині машинного навчання вона зводиться до задачі класифікації часових рядів, а також виявлення та класифікації даних, які попередньо відфільтровано як викиди.

Перед тим як переходити до розгляду класифікації часових рядів, необхідно навести формальне визначення часових рядів та задачі їх класифікації [85]: *Рисунок 17*

Визначення 3: *Одновимірним часовим рядом* $X = [x_1,x_2,\ldots,x_t]$ *називається впорядкована послідовність значень. Довжина ряду дорівнює кількості значень* t.

Визначення 4: *M-вимірний часовий ряд* $X = [X_1 , X_2 , \ldots , X_M]$, *складається з M різних одновимірних часових рядів Xi.*

Визначення 5: *Набір даних* $D = \{(X_1, Y_1),(X_2, Y_2), \ldots ,(X_N , Y_N)\}$ *– колекція пар* (X_i , Y_i) *де* X_i *може бути одновимірним або багатовимірним часовим рядом з відповідним вектором класу* Y_i. *Довжина вектора* Y_i *дорівнює кількості класів K для класифікації де кожен елемент* $i \in [1, K]$ *і дорівнює 1 для кожного з класів з K, і 0 для всіх інших класів.*

Завдання класифікації часових рядів складається з тренування класифікатора на наборі даних D для отримання розподілу ймовірностей за значеннями відповідних класів.

Розглядаючи задачу класифікації часових рядів, можна виділити підходи, які застосовуються найчастіше:

- на основі відстані
- на основі інтервалів
- на основі частоти
- на основі словників
- на основі шарплет перетворень

Основною відмінністю при роботі з часовими рядами в порівнянні з роботою з даними не часової області є інше сприйняття даних [86]. При роботі з табличними даними, малюнками чи текстом, до кожної нової вибірки застосовується однаковий алгоритм навчання, який не враховує послідовність даних. Таким чином, втрачається інформація, що міститься в часовій області (якщо поміняти послідовність даних, то результат лишиться незмінним). Тому, алгоритми, що використовуються для роботи з часовими рядами зазвичай або модифіковані алгоритми класифікації, або їх ансамблі.

Відповідно для виділення ознак з часових рядів також використовуються інші методи, ніж при класифікації даних без часового виміру:

- можуть виділятися глобальні ознаки (використовуючи всі дані)
- локальні ознаки (використовуючи вікна, інтервали, поділ ряду, і т.д.)
- часовий ряд може перетворюватися в вектор певних ознак (наприклад медіана, тренд, розподіл)
- перетворення в інший ряд, без часового виміру (перетворення Фур'є)

Спираючись на такі концепції виділення ознак, розглянемо методи класифікації на основі відстані. Ці алгоритми визначають приналежність поточного ряду до класу, базуючись на метриках відстані. До цих методі відносяться kNN для часових рядів, та використання стандартних методів класифікації використовуючи виділені ознаки відстані [87]. При цьому, kNN що використовується для часових рядів має певну модифікацію. Сам kNN – статистичний непараметричний метод [88], що використовує Евклідову

відстань між даними для виміру схожості. Оскільки така метрика не може використовуватися для часових рядів, так як не враховує зміни часового виміру [89], для їх порівняння використовується динамічна трансформація шкали часу (Dynamic Time Warping) [89]. DWT використовується для порівняння часових рядів, що не нормалізовані між собою.

Якщо для прикладу є 2 часових ряда: $\mathbf{X}=(x_0, \ldots, x_n)$ та $\mathbf{Y}=(y_0, \ldots, y_m)$, то DWT між ними можна порахувати за формулою:

$$DTW(x,y) = \min(\pi)\sqrt{\sum_{(i,j)\epsilon\pi} d(x_{i,}y_j)^2}$$

де $\pi = [\pi_0, \ldots, \pi_K]$ це метрика, що відповідає таким параметрам:

- це список пар індексів $\pi_k = (i_k, j_k)$ з $0 \leq i_k < n$ та $0 \leq j_k < m$
- $\pi_0 = (0,0)$ та $\pi_K = (n-1, m-1)$
- для всіх $k > 0$, $\pi_k = (i_k, j_k)$ так, що:
    - $\pi_k - 1 = (i_k - 1, j_k - 1)$
    - $i_{k-1} \leq i_k \leq i_{k-1}+1$
    - $j_{k-1} \leq j_k \leq j_{k-1}+1$

При цьому цю метрику можна розглядати як вирівнювання часових рядів в часовому просторі таким чином, що евклідова відстань між вирівняними часовими рядами є мінімальною.

Де $d(x_i-y_j)$ – різниця між x_i та y_j. Таким чином для знаходження найближчої точки між x_i та y_j, обчислюється m x n відстаней [90].

Тобто відстань між рядами X та Y використовуючи DWT, обчислюється як квадратний корінь із суми квадратів відстаней між кожним елементом X та його найближчою точкою в Y.

kNN з використанням DWT часто використовують як базову модель, за рахунок простоти у використанні [91]. До недоліків такого підходу можна віднести досить високу обчислювальну складність ($O(MN)$) [91], та недостатню точність з зашумленими даними [91].

3.2.1. Класифікація на основі інтервалів

Використання класифікаторів з ознаками, що виділені з інтервалів ряду. До таких класифікаторів можна віднести Лісовий класифікатор часових рядів (time series forest - TSF), що є адаптацією Random Forest для часових рядів [92]:

- першим кроком часовий ряд розділяється на випадкові інтервали, з випадковими довжинами (для будь-якого часового ряду довжини m існує m (m − 1) / 2 можливих інтервалів, які можна виділити) [93]
- з створених інтервалів виділяються ознаки (медіана, тренд, відхилення) з яких формується вхідний вектор
- будується дерево рішення використовуючи виділені ознаки як вхідні дані
- алгоритм повторюється поки необхідна кількість дерев не буде побудованою

TSF також обчислювально затратний і має складність $O(nlog(n)\cdot m\cdot r)$, де r – кількість дерев в лісі [92].

Нові ряди класифікуються відповідно до класу, який передбачився більшістю дерев. Зазвичай, класифікація на основі дерев показує кращі результати за базові моделі без вчителя (такі як kNN з DWT) [94]. Важливою перевагою використання дерев рішень, є можливість визначити важливість кожної з ознак, що використовуються в передбаченні.

3.2.2 Класифікація на основі словників

Класифікація такого типу базується на перетворенні часових рядів на послідовність векторів, які потім використовуються як «слова» в словнику:

1. часовий ряд розбивається на вікна довжиною w
2. для кожного вікна значення часового ряду перетворюється у вектор («слово») довжиною l

BOSS (bag of symbols)

Працює за тим же методом, що й модель Bag of words [95] для обробки тексту, з тою різницю, що перед тим як кодувати значення часового ряду у вікні, воно перетворюється за допомогою символічної апроксимації Фур'є [96]:

- виконується перетворення Фур'є на вікні

- Дискретизуються перші l гармонік в для формування «слова»
- Гармоніки перетворюються в символи використовуючи розподіл даних на класи (біни)[97], за допомогою алгоритму Multiple Coefficient Binning (показано на Рис.)

З цих «слів» складається словник, таким чином, що якщо одне і те ж слово створюється двома або більше вікнами поспіль, це слово буде зараховано лише один раз (Рис. 3.5). Коли вікно пройшлося по всьому часовому ряду, створення словника завершується і його можна використовувати для тренування будь-якого класифікатора.

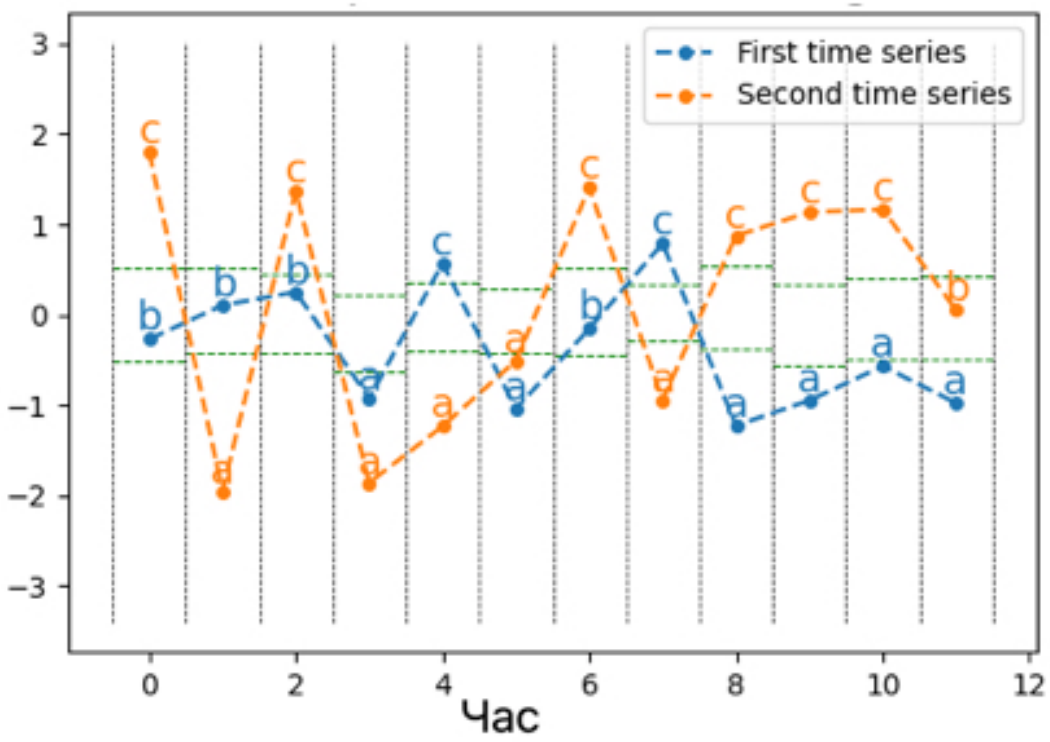


Рисунок 18 Приклад перетворення часового ряду на вектор символів

3.2.2. Класифікація на основі частоти

Одним з самих популярних алгоритмів, що відносяться до цієї групи є Random Interval Spectral Ensemble (RISE) [98], що є модифікацією ліс часових дерев (TSF).

Основна модифікація в порівнянні з TFS полягає в використанні спектральних ознак сигналу, замість статистичних та використанню одного інтервалу для кожного дерева. В якості ознак використовуються:

- Авторегресивні коефіцієнти
- Коефіцієнти автокореляції
- Спектральні ознаки

Таким чином алгоритм має змогу використовувати інформацію з сигналу часового ряду, яка губиться при використанні статистичних ознак сигналу.

RISE має сенс використовувати для довгих послідовностей, наприклад для класифікації звукових послідовностей (для них спектральні характеристики важливіше ніж статистичні при класифікації).

3.2.3. Шейплет (Shapelet) класифікатори

Шейплетами називається частина часового ряду, за допомогою якої можна класифікувати певну частину ряду.

За допомогою шейплетів можна класифікувати різні підкласи в рамках одного часового ряду [99]. Сам по собі шейплет це інтервал, який виділяється в часовому ряді (наприклад в інтервалі [1-4], можна виділити 5 різних інтервалів: [1,2],[2,3],[3,4],[1,2,3],[2,3,4], кожен з яких може бути шейплетом). Приклад шейплета показано на Рис. 3.6.

Шейплет класифікатор визначає наявність k найбільш ймовірних шейплетів в часовому ряді. Наступним кроком визначається відмінність ряду до шейплета, після чого застосовується будь-який алгоритм класифікації. Наявність обчислення відстані від ряду до шейплета показує, що шейплет класифікатори дуже чутливі до наявності шуму.

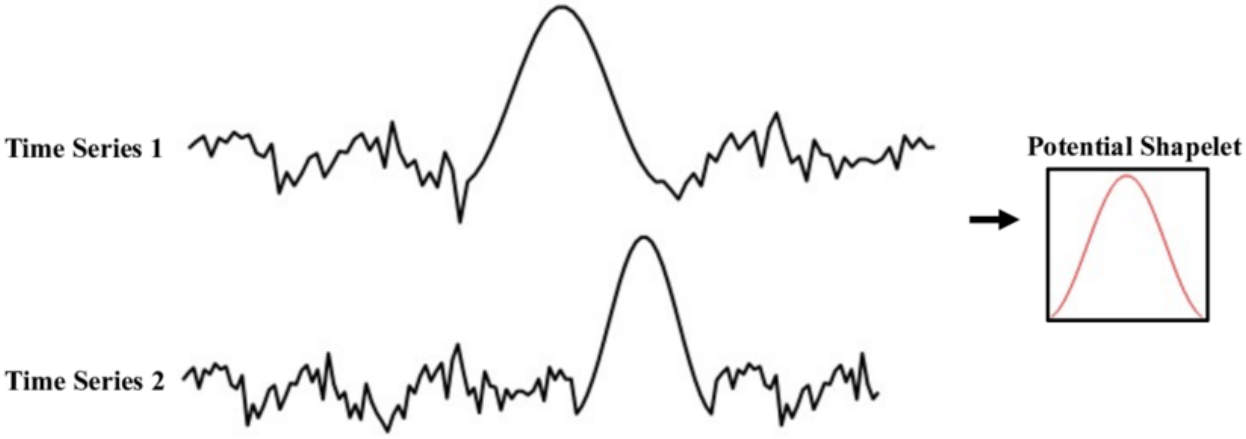


Рисунок 19 Обидва ряди мають локальні сплески, які відрізняються від решти ряду. Ці сплески можна визначити в якості шейплета. [97]

3.3. Методи глибокого навчання для класифікації часових рядів

Методи глибокого навчання також використовуються для класифікацій часових рядів. Як уже було описано в розділі 1, в загальному архітектуру таких мереж можна розділити на такі типи:

- Багато шарові перцептрони
- Згорткові нейронні мережі (CNN)
    - CNN
    - TCN
- Рекурентні нейронні мережі (RNN)
    - LSTM
    - Мережі з відлунням стану (ESN)
- Трансформери

Кожна з архітектур приведених вище має свої плюси та недоліки в якості класифікатора часових рядів.

Однак, перед тим як розглянути детальніше кожну архітектуру, розглянемо основні принципи класифікації часових рядів за допомогою нейронних мереж [85] (Рис. 3.7):

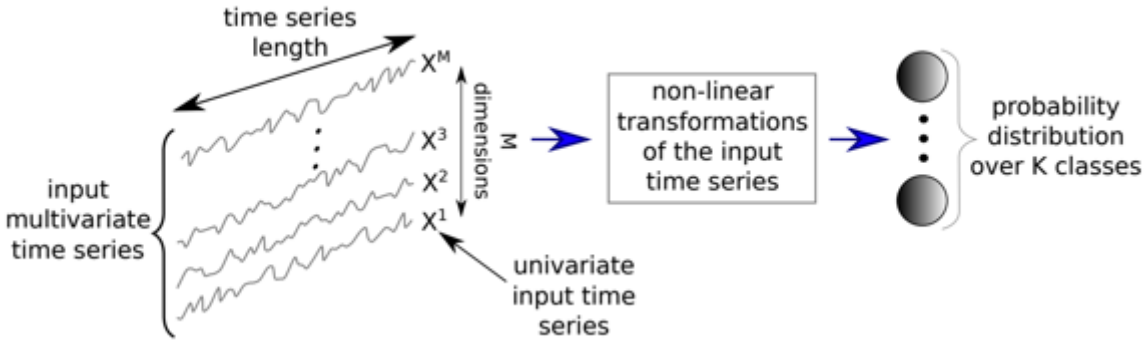


Рисунок 20 Загальна схема роботи нейронних мереж з часовими рядами [85]

Глибока нейронна мережа - це L параметричних функцій, які називаються шарами, де кожен шар вважається поданням вхідної області [100]. Один шар l_i, такий що $i \in 1 \ldots L$ містить нейрони. Шар l_i бере як вихідний сигнал вихідний рівень свого попереднього шару l_{i-1} і застосовує нелінійність (наприклад, сигмоїдну функцію) для обчислення власного виходу. Поведінка цих нелінійних перетворень контролюється набором параметрів θ_i для кожного шару. У контексті DNN ці параметри називаються коефіцієнтами ваги, які пов'язують вхід попереднього рівня з виходом поточного шару. Отже, з урахуванням вхідного значення x, нейронна мережа виконує наступні обчислення для прогнозування класу:

$$f_L(\theta_L, x) = f_{L-1}(\theta_{L-1}, f_{L-2}(\ldots)))$$

де f_i відповідає нелінійності, в шарі l_i. Розглянемо відповідні архітектури зазначених вище мереж:

3.3.1. Багато шаровий перцептрон (multi-layer perceptron)

Це найпростіша архітектура нейронної мережі, яку ще називають повністю зв’язаною (fully connected) [101], оскільки кожен нейрон в шарі li має зв’язки з усіма нейронами попереднього шару. Результатом MLP є послідовно обчислена активація його перцептронів (Рис. 3.8.).

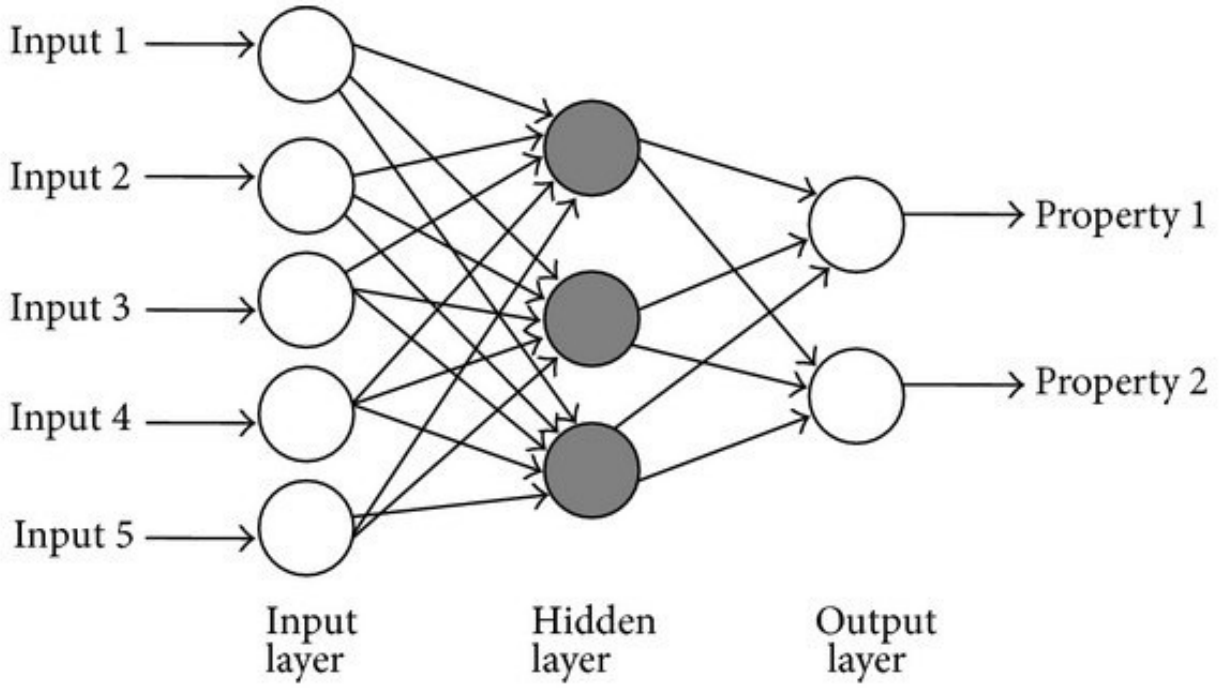


Рисунок 21 Багато шаровий перцептрон (multi-layer perceptron) [102]

За рахунок своєї архітектури, для роботи MLP необхідно:

- кожна вхідна точка ряду має бути представлена у вигляді одновимірного вектора фіксованої довжини
- кожен клас має бути представлений у вигляді одиничного вектору з розмірністю, що дорівнює кількості класів

Для задачі класифікації, в нейронних мережах останній шар як правило дискримінаційний [103]. Він приймає на вході результат активації попереднього шару і на виході показує розподіл ймовірностей для кожного з класів. У більшості випадків використовують шар з кількістю нейронів, яка дорівнює кількості класів з функцією активації softmax. Використання softmax зумовлене 3 особливостями цієї функції:

- сума ймовірностей всіх класів завжди рівна 1
- softmax є диференційованою функцією
- softmax є адаптацією логістичної регресії до мультиноміального випадку

ЇЇ можна записати наступним чином:

$$P_j(X) = \frac{e^{A_{L-1} * w_k + b_k}}{\sum_{k=1}^{K} e^{A_{L-1} * w_k + b_k}}$$

Де P_j – ймовірність того, що X належить до класу j з K класів, w_j (і відповідне зміщення b_j) – вагові коефіцієнти для кожного класу j пов'язані з кожною попередньою активацією в шарі l_{L-1}.

Таким чином процес навчання MLP розбивається на наступні кроки:

- На кожній ітерації результат MLP обчислюється з використанням поточного вхідного вектора
- Результатом є вектор, компонентами якого є ймовірності належності до кожного класу
- Похибка класифікації обчислюється за допомогою функції втрат (loss function)
- Використовуючи градієнтний спуск, ваги оновлюються під час зворотного проходу (backpropagation)

Таким чином, ітеративно роблячи прямий прохід з подальшим зворотним розповсюдженням, ваги моделі оновлюються таким чином, щоб мінімізувати втрати на тренувальних даних.

Недоліком використання MLP для класифікації часових рядів є їх сприйняття ряду як набору непов’язаних між собою векторів. Таким чином втрачається вся додаткова часова інформація.

3.3.2. Згорткові нейронні мережі

Згорткові нейронні мережі – мережі що в основному працюють з зображеннями та багатовимірними часовими рядами. Завдяки використанню фільтрів, такі мережі добре вивчають шаблони та ознаки що повторюються в тренувальному наборі даних.

Згортку можна розглядати як використання фільтра з зсувом по часовому ряду. На відміну від зображень, фільтри що використовуються для часових рядів відображають лише один вимір (час) замість двох вимірів (ширина та висота). Такий фільтр можна розглядати як загальне нелінійне перетворення часового ряду. Наприклад, якщо застосування згортки на одновимірному часовому ряді еквівалентне використанню рухомого середнього з довжиною вікна, яка

дорівнює довжині фільтра. Застосування такого фільтра для загального випадку можна виразити наступною формою:

$$C_t = f\left(\omega * X_{t-\frac{l}{2}:t+\frac{l}{2}} + b\right) | \forall\ t\ \in [1, T]$$

де C - результат згортки, застосованої до одновимірного часового ряду X довжини T з фільтром ω довжиною l, параметром зміщення b та кінцевою нелінійною функцією f (ReLU).

Результат згортки (одного фільтра) на вхідному часовому ряді X можна розглядати як інший одновимірний часовий ряд C, який пройшов процес фільтрації. Таким чином, застосування декількох фільтрів до часового ряду призведе до багатовимірного часового ряду, розміри якого дорівнюють кількості використаних фільтрів.

Оскільки згортки однакові на всьому ряду t ∈ [1, T], то фільтри незалежні від часового виміру.

Згорткові мережі використовують 3 різні типи шарів:

- Згортковий шар (convolutional layer)
- Агрегувальний шар (pooling layer)
- Повноз'єднаний шар (fully-connected layer)
- Шар втрат (loss layer)

Для розуміння роботи згорткових мереж необхідно розглянути принцип роботи кожного шару:

Згортковий шар (Convolutional Layer)

В цьому шарі відбувається накладання фільтра на вхідний сигнал чи малюнок. Вхідний сигнал чи малюнок представлені у вигляді тензора з такими параметрами: довжина часового ряду (чи вікна), кількість каналів сигналу, часові мітки. В якості виходу отримуються тензори з зміненим розміром та високорівневими ознаками, отриманими завдяки згортці.

Згортка визначається набором фільтрів, які є матрицями фіксованого розміру (Рис. 3.9). Результат застосування фільтру - сума добутку кожного елемента фільтра з елементом підматриці на тому ж місці.

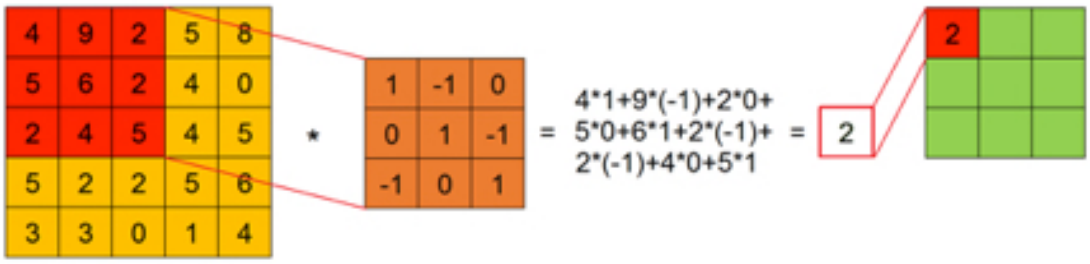


Рисунок 22 Згортковий фільтр

Згортковий шар визначається двома ключовими параметрами – кроком (*stride*) і відступом (*padding*). Крок контролює зсув фільтра. Наприклад якщо крок дорівнює 1, то фільтр накладається з зсувом в одне значення (Рис. 3.10).

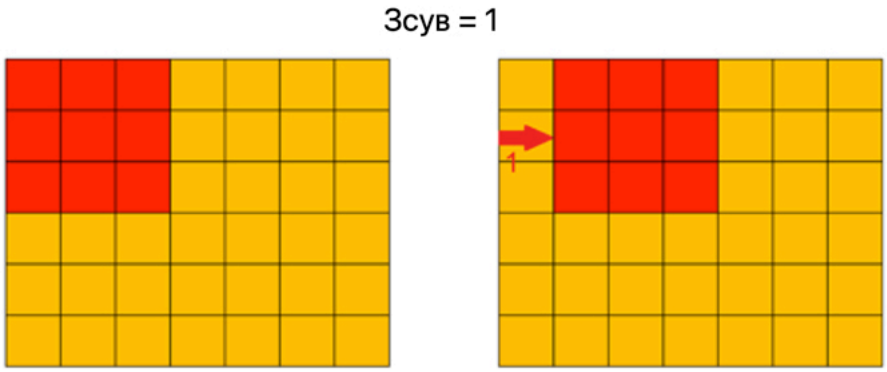


Рисунок 230 Крок згорткового фільтру

Відступ вказує, скільки додаткових стовпців і рядків потрібно додати за межі вхідного сигналу, перш ніж застосовувати фільтр згортки (Рис). Всі додаткові рядки та стовпці заповнюються нулями, щоб не мати впливу на результат згортки. Заповнення використовується, тому що після застосування згортки, початковий розмір даних змінюється, і може стати замалим для фільтра. Таким чином, додаючи зайві рядки та стовпці, зберігається або повільніше зменшується розмір даних (Рис. 3.11).

Padding = 1

0	0	0	0	0	0	0	0
0	1	2	2	3	1	1	0
0	1	4	2	2	7	4	0
0	5	5	6	9	4	1	0
0	4	8	0	4	3	3	0
0	9	0	7	0	4	3	0
0	4	1	0	8	2	1	0
0	0	0	0	0	0	0	0

Рисунок 3.11 Відступ з значенням 1

Агрегувальний шар (*Pooling Layer)*

Метою шару агрегації є агрегація результатів шару згортки, зменшення розмірностей даних, зберігаючи якомога більше інформації. Це корисно також для вилучення домінантних ознак. Існує два типи об’єднання: максимальне об’єднання (Max Pooling) та середнє об’єднання(Average Pooling). Max Pooling працює як придушувач шуму, використовуючи максимальне значення фільтра та відкидаючи шумні активації взагалі. Отже, він, як правило, працює краще, ніж Average Pooling [104]. Перевагою операції агрегації є зменшення вибірки результатів згортки.

Повноз'єднаний шар (*Fully-Connected Layer)*

Повноз'єднаний шар – зазвичай останній шар нейронної мережі як правило дискримінативний. На вхід цього шару подається репрезентація часового ряду після всіх згорток, а на виході отримується ймовірність приналежності до кожного з класів, що є в наборі даних. Зазвичай, так само як і в MLP, цей шар складається з softmax функції [105].

Інколи повноз’єднаний шар використовується додатково крім останнбого шару. Таким чином у мережі збільшується кількість параметрів, що дозволяє їй краще вивчити ознаки.

Для навчання згорткових мереж використовується такий самий процес як і для MLP - прямий прохід з подальшим зворотним розповсюдженням (backpropagation) [106].

Гіперпараметри

Для згорткових, як і для інших нейронних мереж можна визначати багато різних параметрів, проте найважливіші з них [107]:

- Швидкість навчання: визначає, на скільки оновлювати вагу в алгоритмі оптимізації. Використовуються різні значення (зазвичай від 0.1 до 0.001) в залежності від вибору алгоритму оптимізації (SGD, Adam, та інші)
- *Кількість згорткових фільтрів: необхідно шукати баланс, оскільки мала кількість фільтрів не дає змоги мережі навчитися, в той же час як занадно велика кількість не збільшує точність, добавляючи обчислювальної складності*
- Розмір фільтра згортки та початкові значення: маленькі фільтри збирають в основному локальні ознаки, в той час як більші фільтри вивчають більш високорівневу інформацію. Зазвичай фільтри ініціалізуються випадковими значеннями.
- *Метод агрегації (pooling):* Max Pooling прибирає додаткові шуми, тому зазвичай працює краще ніж Average Pooling [107].
- *Початкова ініціалізація ваг: ваги ініціалізуються малими випадковими числами з рівномірним розподілом для уникнення нульового градієнту*
- *Функція активації*: зазвичай sigmoid, relu чи softmax (для внесення нелінійності)
- *Кількість епох навчання:* кількість разів, які весь тернувальний набір проходить через моережу. Кількість має бути достатньою для досягнення якомога меншої різниці між похибкою на тернувальному і валідаційному наборах даних [105].
- Випадання (dropout): метод регуляризації для запобіганню перенвчання. Суть метода в руйнуванні випадкових зв’язків між нейронами з заданою ймовірністю (зазвичай використовується значення 0.5)

Одним з видів згорткових мереж, що добре працюють з часовими рядами є часові згорткові мережі (Temporal Convolution Network) [108]

3.3.3. Часові згорткові мережі

Часові згорткові мережі, або просто TCN, є різновидом згорткових нейронних мереж для завдань моделювання послідовностей, поєднуючи аспекти архітектур RNN та CNN. За рахунок використання довгих послідовностей, TCN показали, що проста згорткова архітектура перевершує складні рекурентні мережі, такі як LSTM, використовуючи довшу ефективну пам'ять.

TCN характеризується такими ознаками [108]:

- Згортки послідновні, тобто інформація не потрапляє з «минулого» в «майбутнє», використовуючи причинно-наслідкові згортки (casual convolutions)
- Мережа може приймати послідовність будь-якої довжини і зіставляти її з вихідною послідовністю тієї ж довжини. TCN має дуже довгу пам’ять (тобто здатність заглянути далеко в минуле, щоб зробити прогноз), використовуючи наявність великої кількості шарів та розширені згортки.

Причинно-наслідкові згортки (casual convolutions) [40][109]

Для досягнення того, що мережа має вихід тієї ж довжини, що і вхідна послідовність, TCN використовує 1D архітектуру повністю згорнутої мережі (FCN), де кожен прихований шар має однакову довжину з вхідним шаром, а додатковий вимір (розмір ядра - 1) доданий, щоб зберегти наступні шари такої ж довжини, як і попередні.

Для того щоб інформація не попадала з «минулого» в майбутнє, використовуються Причинно-наслідкові згортки, де вихідний момент часу t згортається лише з елементами часу t чи раніше.

Фактично, архітектура TCN являє собою **1D FCN + causal convolutions**

Розширені згортки **(Dilated Convolutions)**

Проста причинно-наслідкова згортка може бути приміненою лише на ряд з визначеним розміром. Це ускладнює застосування причинно-наслідкової згортки для дуже довгих послідовностей. Рішення, реалізоване в [110], полягало у використанні розширених згорток, що дозволяють експотенційно збільшувати вхідну послідовність. Для одновимірної послідовності, операція розширеної згортки F на елементі s послідовності визначається за допомогою наступної функції:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i}$$

де d - коефіцієнт розширення, k - розмір фільтра, а s - d · i - це напрямок минулого. Таким чином, розширення еквівалентно введенню фіксованого кроку між кожними двома сусідніми кранами фільтра. Коли d = 1, розширена згортка перетворюється на звичайну [111].

Як видно з опису мережі, така мережа має добре працювати з часовими рядами. Дійсно, у неї є очевидні переваги:

- Паралельність: на відміну від RNN, де передбачення подальших значень відбуваються після завершення попередніх, згортки можуть виконуватися паралельно, оскільки в кожному шарі використовується однаковий фільтр. Отже, як під час навчання, так і під час оцінювання, довга вхідна послідовність може бути оброблена зразу в цілому в TCN**.**
- Стійкі градієнти: на відміну від рекурентних архітектур, TCN має шлях зворотного поширення, інший ніж часовий напрям послідовності [112]. Таким чином, TCN уникає проблеми вибуху / зникнення градієнтів, що є основною проблемою для RNN
- Вхідні дані змінної довжини: Подібно до RNN, які рекурсивно моделюють входи зі змінною довжиною, TCN також можуть приймати входи довільної довжини, використовуючи 1D згортки.

До основних недоліків TCN можна віднести необхідність великої кількості даних для валідації: TCN має використовувати довгі послідовності для валідації, з довжиною, яка приблизно відповідає довжині тренувальної послідовності.

3.3.4. Рекурентні нейронні мережі

Рекурентні нейронні мережі - це вид нейронних мереж, у якому з'єднання між вузлами утворюють граф орієнтований у часі. Така архітектура дозволяє мережі мати внутрішній стан і змінювати свою поведінку з часом (Рис. 3.12). На відміну від нейронних мереж прямого поширення (як CNN), РНМ можуть використовуватися для обробки довільних послідовностей [113].

Різниця в тому, що в RNN мережах кожному нейрону призначається фіксований час.

Кожен з нейронів в прихованому шарі пов'язаний з наступними нейронами цього шару одностороннім з’єднанням, а також з нейронами наступного шару тільки в рамках поточного часу. Вхідні та вихідні нейрони з'єднані лише з прихованими шарами в тій самій часовій точці [113][114].

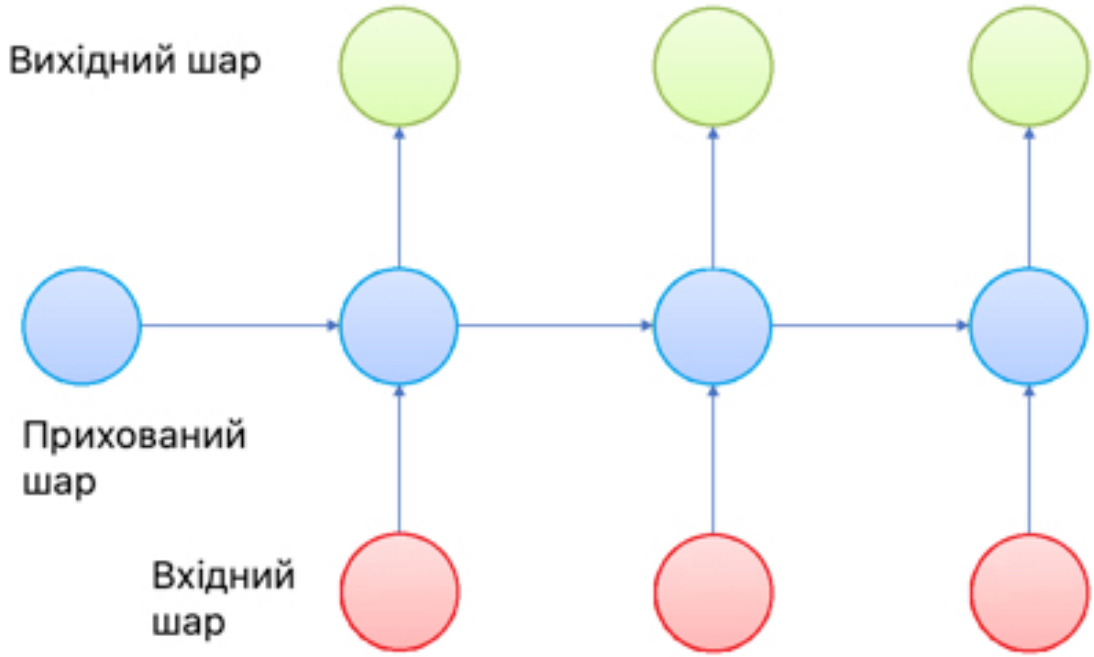


Рисунок 24 Рекурентна нейронна мережа

Оскільки вихід прихованого шару одної мітки часу є частиною входу наступної, активація нейронів обчислюється в порядку часового ряду: у будь-який даний момент часу активуються лише нейрони, що призначені до цієї часової мітки.
Рекурентні нейронні мережі рідко застосовуються для класифікації часових рядів, в основному через три фактори:

- Архітектура призначений більше для прогнозування результату для кожного елемента в часовому ряді, ніж для класифікації.
- При тренуванні на тривалих часових рядах, RNN, як правило, мають проблему зникаючого градієнта, тобто, параметри в прихованих шарах або не сильно змінюються, або такі зміни призводять до нестабільності та хаотичної поведінки мережі.
- Навчання рекурентної мережі неможливо розпаралелити

Враховуючи вищезазначені обмеження, для часових рядів був запропоновано модифікацію рекурентної архітектури: Echo State Networks (ESNs) [115] . ESN були вперше запропоновані в [116] [117]для прогнозування часових рядів у каналах зв'язку. Вони були розроблені для вирішення проблем RNN, усуваючи необхідність обчислювати градієнт для прихованих шарів, що скорочує час навчання, в також допомагає уникнути проблеми зникнення градієнта (Рис. 3.13).
Ці приховані шари ініціалізуються випадковим чином і становлять резервуар: ядро ESN, яке є RNN з розрідженими зв’язками між нейронами. Кожен нейрон в резервуарі створює свою власну нелінійну активацію вхідного сигналу. Ваги всередині резевуару та вхідні ваги не використовують градієнтний спуск під час навчання. Тільки вихідні ваги навчаються за допомогою алгоритму, такого як логістична регресія.

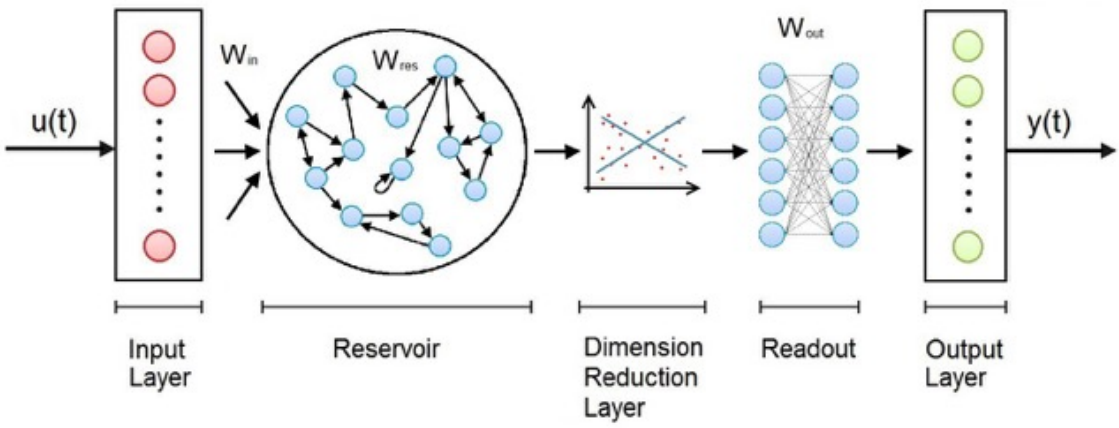


Рисунок 25 Echo State Network [117]

Резервуар (**Reservoir)**

Резервуар являє собою розріджену випадкову RNN. Він підключений до вхідного шару і складається з набору внутрішніх малозв'язаних нейронів та власних вихідних нейронів. В резервуарі є 3 різні типи ваг [117]:

- вхідні ваги між вхідним шаром і внутрішніми
- вихідних ваги між внутрішніми нейронами і виходом;
- ваги зворотного розповсюдження, які з'єднують вихідний сигнал із внутрішніми нейронами.

Всі ці ваги випадково ініціалізовані, однакові для кожної точки часу в ряді і не навчаються.

Як і в RNN, вихід резервуара обчислюється окремо для кожного часового кроку, оскільки вихід часового кроку є частиною входу наступного часового кроку. На кожному часовому кроці обчислюється активація кожного внутрішнього і вихідного нейрона, і отримується вихід для поточного часового кроку.

Великою перевагою ESN є те, що резервуар створює нелінійність вхідних даних, але оскільки тренуються тільки ваги в шарі зчитування, то час навчання залишається низьким невеликим.

3.3.5 LSTM

У RNN є проблеми з використанням короткочасної пам'яті. Якщо послідовність досить довга, їм важко переносити інформацію від попередніх часових кроків до наступних. Це спричиняє потребу у довгостроковій короткочасній пам'яті (LSTM), яка є особливим видом RNN, що здатна вивчати довгострокові

залежності. LSTM [118] маює навички запам’ятовувати інформацію протягом тривалого періоду часу в довгих послідовностях.

LSTM являє собою триступеневий процес, який наведено на Рис, який показує, що модуль LSTM має 3 воріт [119]: ворота забуття (Forget gate), Вхідні ворота (Input gate), Вихідні ворота (Output gate).

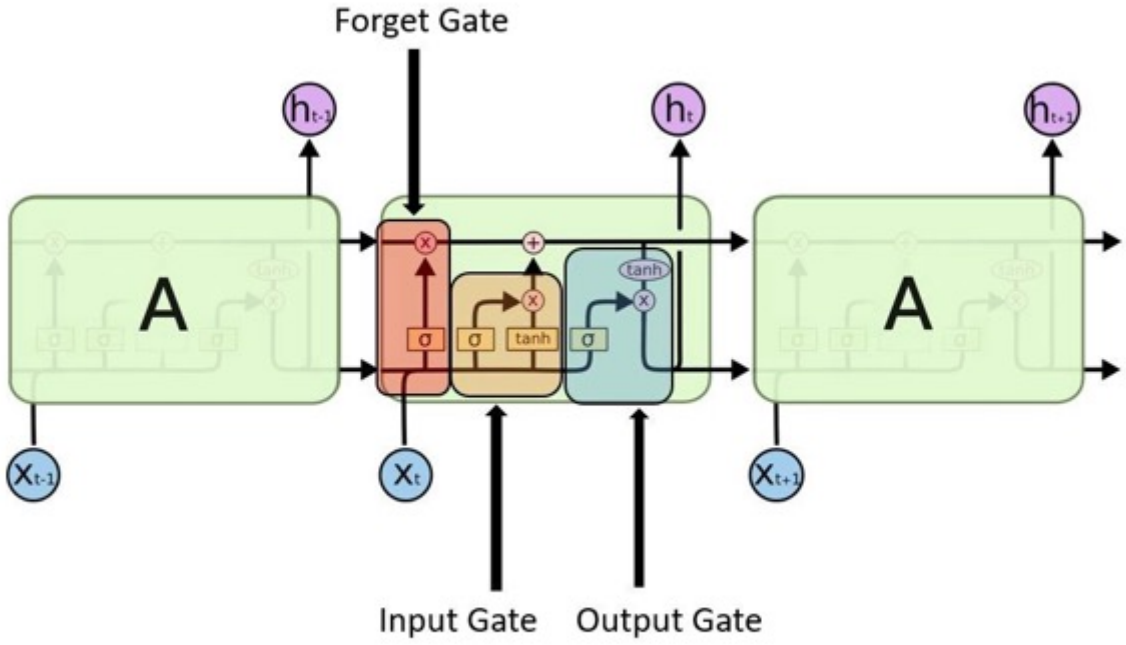


Рисунок 3.14 Комірка LSTM [120]

Основним компонентом LSTM є комірка пам'яті [120] (Рис. 3.14). Вона може підтримувати стан з часом, що включає в себе з попередню пам’ять (вектор стану клітини) та блокуючі елементи. Блоки керування регулюють надходження інформації в пам’ять і з неї.

вектор стану клітини (Cell State Vector) - представляє пам'ять LSTM, і він змінюється через забуття старих станів (Forget gate) і додавання нової пам'яті (Input gate)

Структура воріт [119]:

- Ворота: Sigmoid з подальшим оператором точкового множення.
- Ворота фактично керують потоками інформації в та з пам’яті.

- Ворота управляються за допомогою конкатенації вихідного сигналу з попереднього кроку по часу і поточного входу

ворота забуття Forget Gate:

- Контролюють, яку інформацію необхідно витерти з пам'яті.
- Вирішує, скільки минулого необхідно пам’ятати.

Вхідні ворота (Input Gate)

- Керує тим, яка нова інформація додається до стану комірки з поточного вводу.
- Визначає, скільки з вхідної інформації додано до поточного стану.

Вихідні ворота (Output Gate)

- Умовно вирішує, що виводити з пам'яті.
- Визначає, яка частина поточної комірки потрапляє на вихід

3.3.5. CNN LSTM

Архітектура CNN LSTM [121] передбачає використання шарів згорткової нейронної мережі (CNN) для вилучення функцій на вхідних даних у поєднанні з LSTM для підтримки прогнозування послідовності. CNN LSTM були розроблені в основному для проблем прогнозування часових рядів [122] (Рис. 3.15).

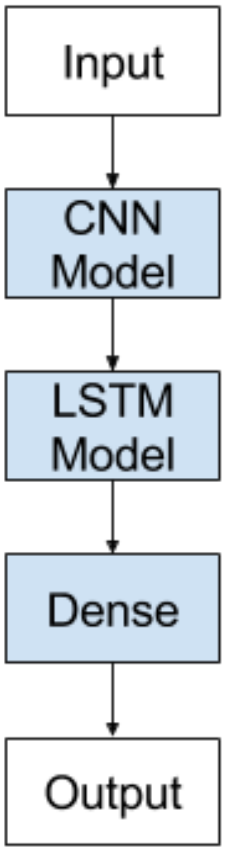


Рисунок 26 Структура CNN LSTM

3.3.6. Трансформери (Transformers)

LSTM обробляють дані послідовно. Теоретично, завдяки прихованому стану важлива інформація може поширюватися у нескінченно довгих послідовностях. Однак на практиці це не так. Через проблему зникаючого градієнта LSTM забуває попередні дані.

З іншого боку, трансформери зберігають прямі зв'язки з усіма попередніми мітками часу, дозволяючи пам'ятати набагато довші послідовності. Однак це спричиняє іншу проблему: модель буде безпосередньо очікувати довгу послідовність вхідних даних. Для того, щоб відфільтрувати важливе від неважливого, трансформери використовують алгоритм, який називається механізм уваги.

Механізм уваги (Attention Mechanism) [123]

Механізм уваги має зосереджуватися тільки на найважливіших підмножинах довільно довгих послідовностей, які мають значення для виконання даного завдання.

По суті, модель повинна вирішити, які деталі попередніх вхідних даних мають значення для визначення поточного [124]. Механізм уваги визначає кожен новий вхідний елемент по відношенню до всіх інших попередніх елементів, проте найбільшу вагу має поточний вихідний елемент:

- Створення вектора запитів, ключів та значення (query, key, value): кожен вхідний елемент генерує запит, ключ і значення. Запит поточного елементу порівнюється з ключами всіх інших елементів, щоб визначити їх відповідність щодо поточного. Під час навчання модель поступово засвоює ці три ваги, на які множиться кожен елемент, щоб сформувати відповідний ключ, запит та значення [125].
- Розрахунок оцінки уваги: це розрахунок міри відповідності між поточним елементом та будь-яким іншим елементом, який раніше вже був у послідовності. Він обчислюється шляхом обчислення точкового добутку між вектором запиту поточного елемента та ключовим вектором оцінюваного маркера ($query_i$ x key_j) [126]. Потім значення оцінки уваги передається через функцію softmax. На цьому кроці оцінку уваги можна розглядати як відсоток від загальної уваги, що приділяється елементу в послідовності.
- Визначення поточного елементу: для цього кожен вектор значеннь множиться на його оцінку уваги. Це зберігає вихідне значення незмінним, але масштабує загальний вектор відповідно до його відносної важливості для поточного елемента. Потім, усі масштабовані значення підсумовуються для отримання вектора поточного елемента.

$$Attention(Q, K, V) = softmax(QK)V$$

Таким чином використання механізму уваги разом з LSTM для довгих послідовностей може значно покращити результат в порівнянні з використанням тільки LSTM [127].

3.3.7. Трансферне навчання

Визначення 3: *Трансферне навчання глибоких нейронних мереж - це процес навчання базової мережі на вихідному наборі даних та завдання, а потім*

передача вивчених функцій (ваги мережі) у другу мережу для навчання на цільовому наборі даних та завданні .

Трансферне навчання іноді плутають з підходом адаптації доменів [128]. Основна відмінність від останнього методу полягає в тому, що у випадку трансферного навчання, модель донавчається на вихідних та цільових наборах даних [14]. Основною метою під час навчання є мінімізація розбіжностей між екземплярами джерела та цілі.

В [129] показано використання можливості трансферного навчання для підвищення точності задачі класифікації часових рядів. У цій роботі автори спроектували CNN із механізмом уваги (attention mechanism). Перед остаточним тренуванням моделі на цільовому наборі даних, модель спочатку попередньо навчається на декількох наборах вихідних даних, які відрізняються від цільового набору даних, що обмежує вибір вихідного набору даних лише одним. Крім того, на відміну від [129], береться заздалегідь розроблена модель глибокого навчання, не змінюючи її та не додаючи регуляторів. Це дозволило пов’язати покращення точності виключно з функцією навчання.

Можна детальніше розглянути процес трансферного навчання для часових рядів. В роботі [130] описується процес навчання однакової мережі на 85 наборах даних, що в результаті створює 85 різних нейронних мереж. Єдина різниця між цими 85 архітектурами нейронних мереж полягає у вихідному шарі. Решта шарів мають однакову кількість параметрів, але з різними значеннями. Насправді останній шар, який є кінцевим класифікатором (softmax), залежить від кількості класів у наборі даних. Таким чином, враховуючи вихідний набір даних D_s і цільовий набір даних D_t, спочатку мережа навчається на наборі даних D_s. Потім останній шар замінюється іншим шаром (також softmax), кількість нейронів якого дорівнює кількості класів у цільовому наборі даних $_{Dt}$.

Параметри доданого шару softmax ініціалізуються випадковим чином за допомогою рівномірної ініціалізації Глорота [131]. Потім ця нова мережа перетреновується на наборі даних D_t. Існує два варіанта дотренування мережі на новому наборі даних – дотреновувати тільки останній шар, чи дотреновувати

всю мережу. В [132] показано, що зазвичай дотреновування всієї мережі дає значно кращі результати ніж дотреновування тільки нового шару.

Однією з основних проблем, пов'язаних з трансферним навчанням, є вибір початкового набору даних. У роботі [133] було продемонстровано, що модель, навчена на певному наборі даних, не дасть оптимального результату, якщо розподіл вхідних даних в цих наборах буде різним. Таким чином, якщо розглядати задачу HAR, то для використання можливостей трансферного навчання, необхідно вибирати набори даних в рамках однієї активності, таким чином, щоб було співпадіння розподіла зібраних даних.

В якості застосування трансферного навчання для визначення дихання, було використано публічний набір даних [134], який складається з записів ЕКГ довжиною від 7 до 10 годин, а також містить проанотовані відрізки приступів апное. Даний набір даних було використано в якості початкового набору для задачі визначення типу дихання. Оскільки приступ апное нерозривно пов'язаний з зміною дихання, такий набір даних може бути використаний для трансферного навчання. З заміною тільки останнього шару моделі, і використовуючи архітектуру мережі, описану в розділі 4, на зібраному наборі даних була досягнута точність класифікації типу дихання 76%. Враховуючи що без використання трансферного навчання, точність склала 88%, можна побачити, що для трансферного навчання необхідно використовувати максимально схожі набори даних. В іншому випадку кінцева точність моделі може зменшуватися. З іншого боку, така технологія може використовуватися в разі наявності вхідних даних в кількості, недостатній для навчання мережі з нуля, і обмеженої можливості збору нових даних.

3.4. Порівняння методів машинного навчання для класифікації часових рядів

Для порівняння ефективності методів машинного навчання часто використовуються кілька наборів даних. Найбільш відомими наборами даних, що використовуються для порівняння класифікації часових рядів, є набори даних, зібрані Університетом Каліфорнії [135]. Вони включають в себе 120

різних наборів даних, що складаються з часових рядів різної довжини та розмірності. Процес порівняння різних методів на цих наборах даних детально описано в [136]. В результаті порівняння в [136] виявлено, що для часових рядів, які містять шум, найбільш точними виявились згорткові нейронні мережі. Також, непоганий результат показали LSTM та TCN. Щоб перевірити наскільки такі методи точні для HAR, порівняємо їх точність для набору даних, зібраних в розділі 4. Результати точності класифікації можна побачити в таблиці 2. Для порівняння були використані вхідні дані, що пройшли однакову попередню обробку. Також слід зазначити, що для всіх мереж був використаний однаковий параметр попередньої зупинки (early stopping), що дало змогу не залежати від кількості епох навчання.

Таблиця 2

Порівняння моделей класифікації часових рядів

	Зібраний набір даних		Набір даних ShakeGestureWiimoteZ з UCR [135]	
	F1	Testing Accuracy	F1	Testing Accuracy
TSF	0.53	0.47	0.7	0.75
CNN	0.83	0.836	0.92	0.96
LSTM	0.78	0.744	0.84	0.87
TCN	0.68	0.612	0.93	0.95

Таким чином, з таблиці можна зробити висновок, що згорткові мережі та їх модифікації найкраще підходять для класифікації рухів людини. Також видно, що вони значно краще працюють з зашумленими даними. Відмінність зібраного набору даних від набору даних з UCR в основному в тому що набір даних з UCR більш очищений.

З таблиці 2 також видно, що всі моделі краще працювали з більш чистим набором даних UCR, і трохи гірше з реальними даними, отриманими з сенсорів.

Висновки до розділу 3

1. Розглянуто методи машинного навчання для класифікації часових рядів. Зокрема розглянута таксономія методів машинного навчання і окремо розглянуті статистичні методи та методи глибокого навчання. Оскільки статистичні методи менш гнучкі у вивченні патернів часових рядів, вони як наслідок рідше використовуються для задачі їх класифікації. Однак можуть використовуватися у попередній обробці даних для отримання додаткових ознак.
2. Більш детально розглянуто різні архітектури нейронних мереж що використовуються для класифікації часових рядів. Основний акцент зроблено на згорткових, рекурентних мережах та їх модифікаціях (таких як LSTM чи трансформери). Проаналізовано можливість використання різних архітектур нейронних мереж для визначення різних типів людської діяльності.
3. Проаналізовано можливість використання трансферного навчання для розпізнавання типу дихання людини на прикладі зібраного набору даних. Оскільки часові ряди мають досить багато унікальних ознак, то на прикладі визначення типу дихання за допомогою попередньо навченої моделі на наборі даних з ЕКГ, продемонстровано, що трансферне навчання може частково застосовуватися для класифікації часових рядів у випадку коли обмежена можливість збору додаткових даних.
4. Проведено аналіз використання розглянутих раніше архітектур нейронних мереж на зібраному наборі даних для класифікації типів диханя.

Розділ 4 Використання методів HAR для розпізнавання типу дихання

4.1. Загальна архітектура моніторингу дихання

В попередніх розділах фокус був зосереджений на отриманні, фільтрації та класифікації даних, отриманих з сенсорів. Однак, для збору цих даних необхідно створити систему з відповідною інфраструктурою. Така система має відповідати певним вимогам:

- Безпекові – потреба в забезпеченні доступа до даних, неможливості неавторизованим користувачам отримувати доступ до даних користувача; можливості розмежування прав доступу для користувачів та додаткових застосунків.
- Інфраструктурні – можливість горизонтального масштабування та незалежного розгортання компонентів системи як невід'ємна складова стійкості до відмов.
- Автономні – система має мати змогу функціонувати (виконувати критичні функції) без доступності до зовнішньої мережі
- Мобільні – система має мати зможу певний час функціонувати без постійного живлення

Зважаючи на структуру системи та вимоги до автономності і мобільності – най оптимальнішим способом комунікації між компонентами системи є бездротова взаємодія в рамках однієї мережі.

Безпровідні мережі можна розбити на три основні категорії: Wireless Wide Area Networks (WWAN), Wireless Local Area Networks (WLAN) та Wireless Personal Area Networks (WPAN) [137].

WWAN мережі використовуються в задачах, які потребують передачі даних в мережі на великі відстані. Найпоширенішою реалізацією є широкосмуговий стільниковий зв'язок. Для інтеграції в таку мережу туманні вузли повинні мати стільникові модулі. Широкосмуговий канал дозволяє передавати дані з швидкістю понад 1Gbps, а в випадку 5G навіть 10Gbps, проте його підтримка є досить енергозатратною. Насьогодні розробляються енергоефективні

альтернативи для використання в системах інтернету речей (наприклад, LPWAN), які пропонують різні варіанти балансу між радіусом покриття, пропускною здатністю та енергоефективністю.

WLAN мережі покривають обмежену територію однієї будівлі або їх невеликої групи. Зазвичай, в даній групі розглядаються стандарти сімейства Wi-Fi [138]. Пропонується широкий вибір реалізацій протоколів, зокрема покликані вирішити задачі розподілених мереж інтернету речей. До недоліків варто віднести низький рівень впровадження енергоефективних версій стандарту індустрією.

WPAN мережі характеризуються невеликим радіусом дії та споживанням енергії. В поєднанні з невисокою ціною комунікаційних модулів це робить їх оптимальним вбором для мереж смарт-сенсорів. Існує велика кількість широко впроваджених протоколів: Bluetooth, BLE [139], тощо, кожен з яких має переваги для окремих сценаріїв використання. Окремою підкатегорію WPAN є мережі зв'язку на невеликих відстанях (англ. Near Field Communications, NFC). Дана група протоколів характеризується дуже високою енергоефективністю і зазвичай працює в режимі «маячка», який розсилає повідомлення під'єднаним вузлам або навіть за їх відсутності в пуш-режимі. ЇЇ найпоширенішими протоколами є BLE та RFID [140].

Таблиця 3

Технології безпровідної комунікації між сервісами

Технологія	Пропускна здатність	Енергоефективність	Радіус дії
4G, 5G	>1Gбіт/с	низька	Покриття оператором
LPWAN	<200Кбіт/с	середня	1-10км
Wi-Fi	>1Gбіт/с	низька	<100м
Bluetooth	2.1 Мбіт/с	середня	<100м

BLE	1 Мбіт/с	висока	<50 м
NFC	424 Кбіт/с	висока	< 20 с

Виходячи з порівняння наведеного в Таблиця 3 порівняння, для міжсенсорної комунікації пропонується використовувати Bluetooth, при цьому смартфон користувача виступає шлюзом, котрий забезпечує інтеграцію з зовнішньою мережею через широкосмуговий стільниковий зв'язок та Wi-Fi.

Відповідно, загальна архітектура системи складається з натільних сенсорів, що з'єднуються з мобільним пристроєм, використовуючи Bluetooth, коли прилад є поблизу (в разі відсутності мобільного приладу в зоні досяжності, дані зберігаються на карті пам'яті). Після передачі даних вони проходять попередню обробку, і після цього класифікацію. Далі дані пересилаються в хмару, використовуючи API, де зберігаються в базу даних, і можуть бути в подальшому використаними для аналізу.

Кожен з компонентів системи виокремлюється в окремий «сервіс», котрий відображає певний процес або його самодостатню частину і використовує стандартизовані API для комунікації з іншими сервісами. Даний підхід визначається як Сервісно-Орієнтована Архітектура (СОА) [141]. Окремі сервіси інкапсулюють виконання своїх функцій від іншої частини системи. Даний підхід підпадає під визначення мікросервісної архітектури (англ. Microservice Architecture, MSA) [142]. Завдяки використанню мікросервісних архітектур отримуються наступні переваги:

- Використання різних технологій та провайдерів технологій
- Стійкість до відмов (кожен компонент незалежний від іншого)
- Масштабованість

Існує три класи додатків, які зазвичай реалізуються за допомогою обробки потоку даних:

1. Орієнтовані на події. Це еволюція мікросервісів. Вони спілкуються через журнали подій замість викликів REST і зберігають дані програми як

локальний стан, замість того, щоб записувати їх і читати із зовнішнього сховища даних, наприклад, реляційної бази даних або сховища ключів-значень [143]. Вони мають кілька переваг у порівнянні з транзакційними програмами або мікросервісами. Локальне збереження поточного стану забезпечує дуже хорошу продуктивність у порівнянні з читанням та запитом запитів до віддалених сховищ даних. Масштабування та відмовостійкість обробляються у вигляді потоку, і за допомогою журналу подій як джерела вхідних даних, стан програми зберігається і може бути повністю відтворений.

2. Конвеєр даних. Традиційним підходом до синхронізації даних у різних системах зберігання є періодичні завдання ETL [144]. Однак вони не відповідають вимогам до пропускної здатності в багатьох випадках. Альтернативою є використання журналу подій для розповсюдження оновлень. Оновлення записуються та розповсюджуються в журналі подій. Споживачі журналу записують оновлення до сховищ даних [145]. Залежно від використання, передані дані, інколи, доводиться нормалізувати, збагатити за допомогою зовнішних даних або агрегувати перед тим, як сховища даних їх приймуть.

3. Аналітичні додатки. Замість того, щоб чекати періодичного запуску, програма потокової аналітики постійно поглинає потоки подій та оновлює свої результати, включаючи останні події з низькою затримкою. Це схоже на методи технічного обслуговування систем баз даних, які використовуються для оновлення матеріалізованих пердставлень. Як правило, потокові програми зберігають свої результати у зовнішньому сховищі даних, яке підтримує ефективні оновлення, наприклад, у базі даних або сховищі ключ-значення. Оновлені результати програми потокової аналітики можна використовувати для візуалізації звітів.

Для обробки постійних нових даних, які надходять з сенсорів, необхідно використовувати архітектуру системи, яка підходить для обробки потокових даних, що підпадає під випадок конвеєру даних.

У порівнянні з традиційною архітектурою big data, архітектура потокового передавання даних, як правило, зовсім інша. Дані обробляються у вигляді потоків протягом усього процесу, а не тільки використовуючи пакетну обробку. Дані, що обробляються потоковою обробкою, безпосередньо надсилаються споживачам у вигляді повідомлень. Таким чином однією з переваг можна назвати відсутність традиційного ETL процесу при малому часі обробки потокових даних. З недоліків такого підходу можна назвати відсутність пакетної обробки, тому відтворення даних та історична статистика не можуть бути добре підтримані. Для офлайн-аналізу підтримується лише аналіз у вікні.

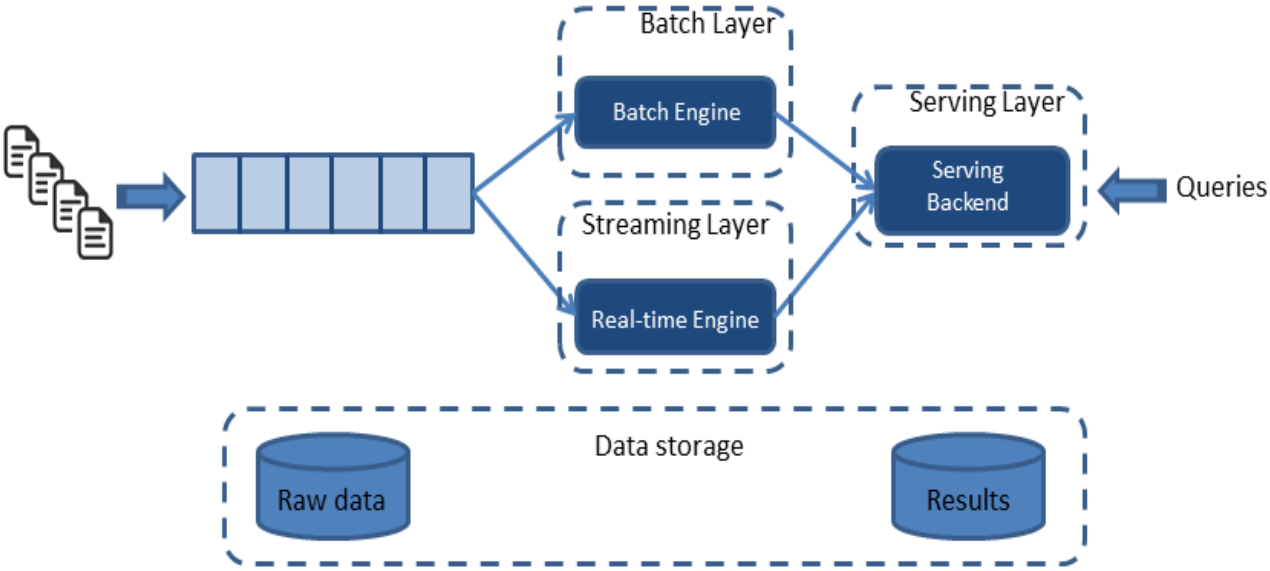


Рисунок 4.1 Lambda архітектура [146]

Lambda-архітектура [147] (Рис. 4.1) є ключовою архітектурою в системах big data. Більшість архітектур - це в основному лямбда-архітектура або архітектури, засновані на її варіантах. Канал передачі даних Лямбди розділений на дві гілки потокового передавання в режимі реального часу та пакетної обробки. Для того, щоб забезпечити ефективність обробки потокового каналу, додатковий розрахунок є основним допоміжним посиланням, тоді як рівень пакетної обробки виконує повні обчислення даних, щоб забезпечити їх остаточну

узгодженість [148]. Отже, крайній шар лямбда архітектури має шар реального часу та офлайн-шар. До переваг лямбда архітектури можна віднести обробку даних як в режимі реального часу, так і в режимі офлайн (аналітичному режимі).

Основним недоліком такої архітектури можна назвати дуплікацію даних. Хоча офлайн-рівень і потік у реальному часі стикаються з різними сценаріями, їх внутрішня логіка обробки однакова, тому існує багато дублікатів.

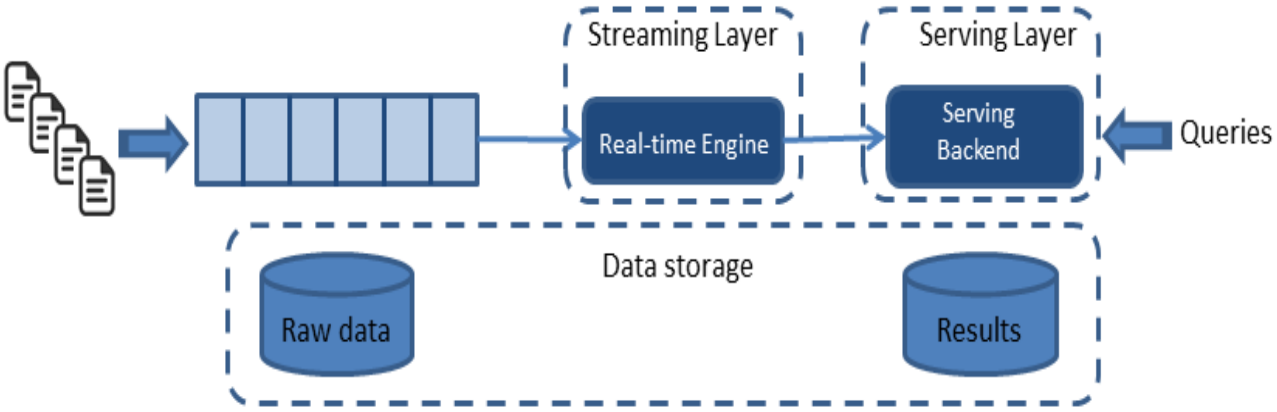


Рисунок 4.2 Kappa архітектура [149]

Архітектура Kappa [150] (Рис. 4.2) оптимізована на основі Lambda, поєднує обробку даних в реальному часі та використовує чергу повідомлень при пакетній обробці. Kappa використовує обробку потоків як основу, але дані зберігаються на рівні озера даних (data lake) [151]. Коли потрібна офлайн-аналітика та різні інші багаторазові обчислення, дані з озера даних можуть знову передаватися через чергу повідомлень.

Таким чином перевагою архітектура Kappa є відсутність дублікатів та надлишковості архітектури Lambda. Недоліком можна назвати складність в реалізації, особливо для частини відтворення даних.

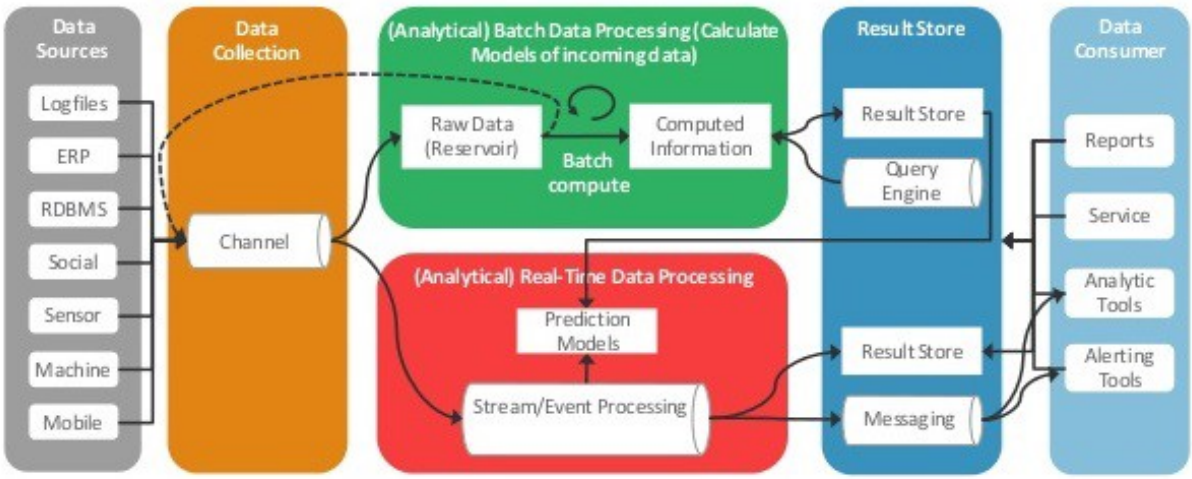


Рисунок 4.3 Об’єднана Lambda та Kappa архітектура [149]

Kappa та Lambda в основному зосереджені на потоковій обробці даних, тоді як об’єднана архітектура [149] [152] (Рис. 4.3), поєднує машинне навчання та обробку даних. По суті, об’єднана архітектура все ще базується на Lambda, але до неї було додано потокову обробку. Додано новий рівень машинного навчання - дані надходять і передаються через канал даних, додається нова навчальна частина моделі, яка використовується в потоковому рівні. Цей потоковий рівень використовує модель і постійно навчає модель.

До переваг такої архітектури можна віднести набір рішень, що поєднують аналіз даних та машинне навчання.

Недоліком також можна вважати складність в реалізації. Що стосується компоненту машинного навчання, то все дуже залежить від того, яке використовується програмне забезпечення.

Традиційним підходом, який призначений для обробки потоку даних, включаючи дані з сенсорів, що генеруються IoT, є модифікація лямбда-архітектури (Рис. 4.4), яка об'єднує аналіз даних з сенсорів в режимі реального часу та історичних даних в рамках єдиної системи [153]. Потоки даних надходять до черги повідомлень (або іншого джерела даних), де відбувається їх обробка:

1. За допомогою шару швидкісної обробки: Результати надаються в реальному часі синхронно (за допомогою відповідей на виклики API) або асинхронно (за допомогою спеціального API).
2. За допомогою шару пакетної обробки: необроблені дані зберігаються в озері даних, а потім обробляються та зберігаються в сховищі даних, і аналізуються за розкладом або на вимогу.

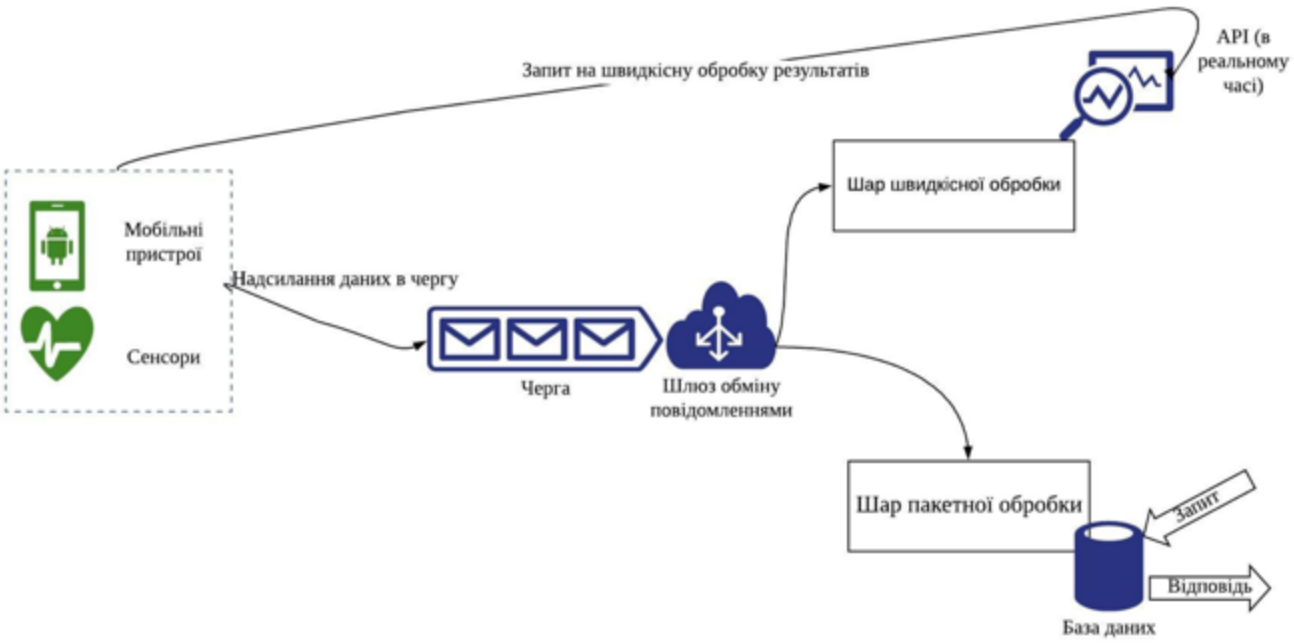


Рисунок 4.4 Обробка даних з сенсорів за допомогою лямбда-архітектури

Використовуючи можливості сучасних мобільних пристроїв, вдалося зробити варіант лямбда-архітектури з використанням шлюзів (в даному випадку смартфонів) у якості обчислювальних вузлів (Рис. 4.5). Такий варіант дозволяє використовувати мобільні пристрої не тільки для збору та передачі даних, а також для обробки даних в реальному часі, що надає такі переваги:

- Компонент швидкого аналізу даних переноситься з хмари на прикінцевий рівень мережі. При цьому результати обробки продовжують відправлятися в хмару для постійного збереження. На додачу до зниження мережевої затримки, оптимізації обсягу переданих в хмару даних, зниження безпекових вимог до баз даних (адже тепер зберігаються результати обробки, витік яких є безпечнішим з точки зору захисту

приватності користувача), даний підхід знижує вимоги доступності хмарного компонента системи, дозволяючи хостити її на спотових серверах, котрі мають знижену динамічну вартість та понизити вимоги стійкості вузла до відмов, що може дозволити відмовитися від надлишкової інфраструктури [154].

- Дані передаються в хмару в попередньо обробленому стані, усуваючи необхідність в озері даних і істотно зменшуючи необхідність в додатковому аналізі

Такий метод розподілу обчислювальних ресурсів називається крайові або прикінцеві обчислення (Edge computing) [155] і визначається як будь-які обчислювальні та мережеві ресурси між джерелом даних та центром їх обробки та збереження (хмарним чи локальним) як вузловий обчислювальний вузол.

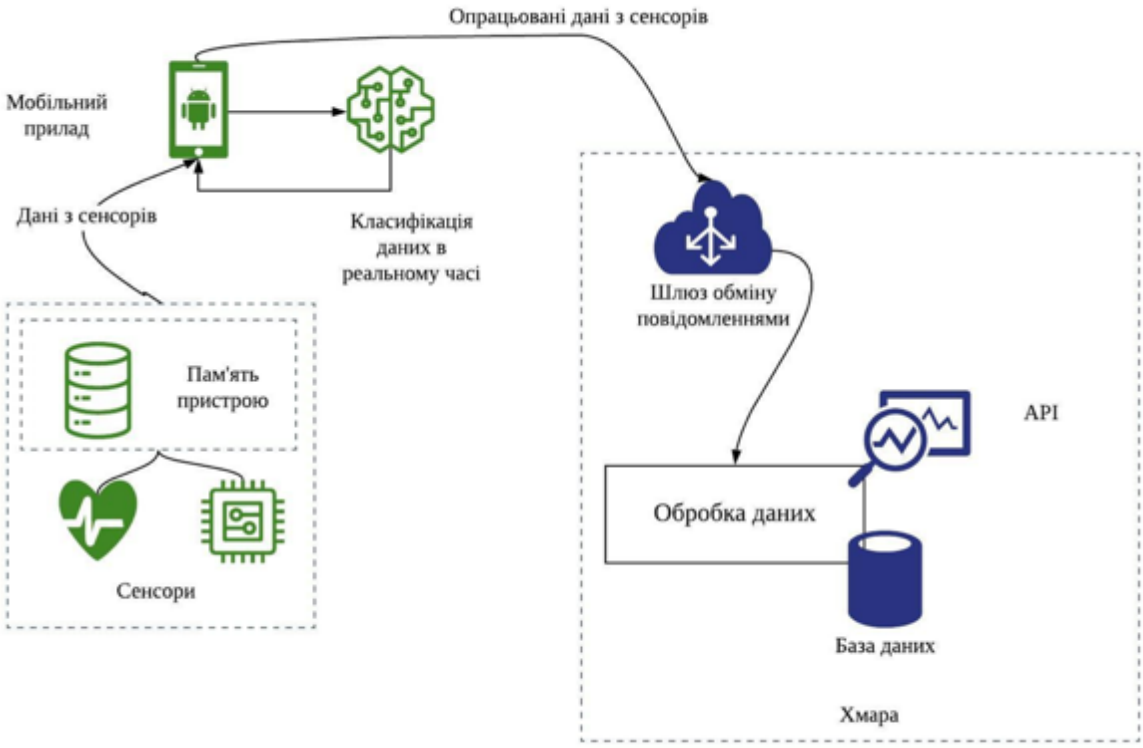


Рисунок 4.5 Архітектура сервісу збору даних

4.2. Прилад

Для перевірки гіпотез був розроблений прототип приладу визначення стану пацієнта на основі частоти його дихання.

Виходячи з гіпотези, що під час дихання рух грудної клітки є комбінацією кількох базових рухів, було прийнято рішення створити систему натільних сенсорів, які кріпляться до одягу пацієнта, і дослідити визначення людської активності з багатомодального сигналу, різних типів сенсорів.

Спроектований пристрій складається з натільної частини, що кріпиться до грудної клітини і складається з наступного набору сенсорів розпізнавання рухів грудної клітки та живота користувача, емпірично підібраних для коректної роботи за різних положень тіла користувача (Рис. 4.6):

1. Комбінований трьохосний акселерометр та гіроскоп. Згідно з дослідженнями, проведеними в [63] оптимальним розташуванням даного типу сенсора є розміщення збоку знизу грудної клітки, адже ця частина тіла має максимальну амплітуду під час дихання.
2. Двох гнучких датчиків деформації, котрі згинаються при русі живота і дозволяють відслідкувати діафрагмальне дихання
3. Карти пам’яті і модуля bluetooth

Наведені сенсори підключено до центрального вузла в вигляді системи-на-чіпі, котрий забезпечує передобробку та передачу сигналу далі на мобільний пристрій. Для побудови прототипів було вибрано SOC на базі ESP32 [156].
Такий прилад фіксує рух грудної клітини і записує покази на карту пам’яті. Якщо він підключений до мобільного пристрою за допомогою bluetooth, то також передає дані у розроблений додаток. Обробка даних відбувається на мобільному пристрої.

Рисунок 4.6 прилад моніторингу дихання

Щоб перевірити пропоновані рішення і гіпотези, було зібрано і сформовано набір даних. З цією метою було розроблено додаток для Android (Рис. 4.8), який збирає дані з приладу, прикріпленого до грудей користувача, і передає дані на смартфон, використовуючи Bluetooth з частотою дискретизації 30 Гц (30 значень в секунду).

Для використання моделі на мобільному пристрої використовувався TensorFlow Lite [157]. Процес використання моделі для отримання прогнозів з нових даних називають *inference*. Щоб використати модель TensorFlow Lite у режимі *inference*, її потрібно запустити через інтерпретатор. Інтерпретатор використовує статичне впорядкування графів та спеціальний розподіл пам'яті, щоб забезпечити мінімальне навантаження, ініціалізацію та затримку виконання.

Модель TensorFlow Lite генерується з моделі TensorFlow [158][159] за допомогою конвертораю Створюється файл .tflite, що являє собою оптимізований формат FlatBuffer [160].

Процес *inference* зазвичай виконує таку послідовність кроків (Рис. 4.7):

1. Loading a model Загрузка моделі. Модель що містить граф виконання моделі завантажується в пам’ять з файлу .tflite

2. **Трансформація даних.** Вхідні дані трансформуються таким чином, щоб вони відповідали формату, який очікує модель. Зазвичай така трансформація називається попередньою обробкою даних.

3. **Inference.** Цей крок передбачає використання API TensorFlow Lite для запуску моделі.

4. **Отримання результатів.** В більшості випадків, модель повертає список ймовірностей, які необхідно трансформувати таким чином, щоб зпівставити ймовірності з відповідними вхідними категоріями.

TensorFlow API існує для більшості мобільних та вбудованих платформ, таких як Android, iOS та Linux, різними мовами програмування, що дозволяє використовувати моделі, конвертовані в TensorFlow Lite на різних платформах.

У більшості випадків API TensorFlow Lite створено таким чином, щоб забезпечити найкращу продуктивність, інколи за рахунок простоти використання.

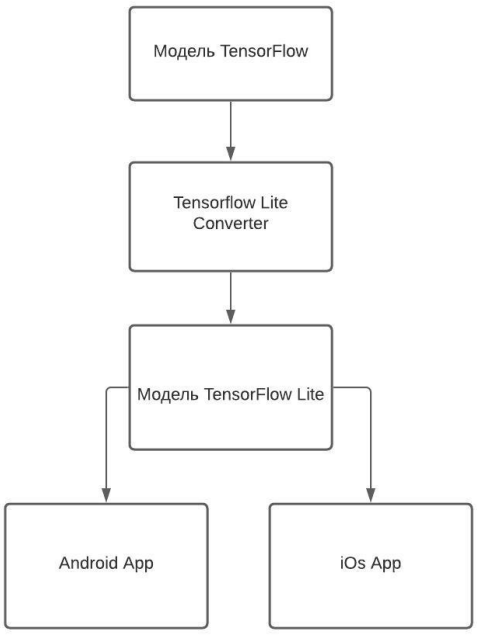


Рисунок. 4.7 Алгоритм генерації моделі TensorFlow Lite

4.3. Збір та розмітка даних

Щоб отримати розмічений набір даних, покази приладу було зібрано з 8 користувачів різного віку та фізичної форми. Для визначення різних типів дихання, користувачі перед вимірами робили відповідні рухи. Типи дихання, які записувалися включають в себе: спокійне дихання, дихання після віджимання, дихання під час бігу, глибоке дихання, швидке дихання, дихання з затримкою. Прилад закріплений над ребрами під грудьми (як показано на Рис. 1). Використовуючи скотч-стрічку і пробуючи кілька різних місць на грудях, було з'ясовано, що це положення дає найбільшу амплітуду руху грудної клітини і, як наслідок, найчистіший сигнал дихання.

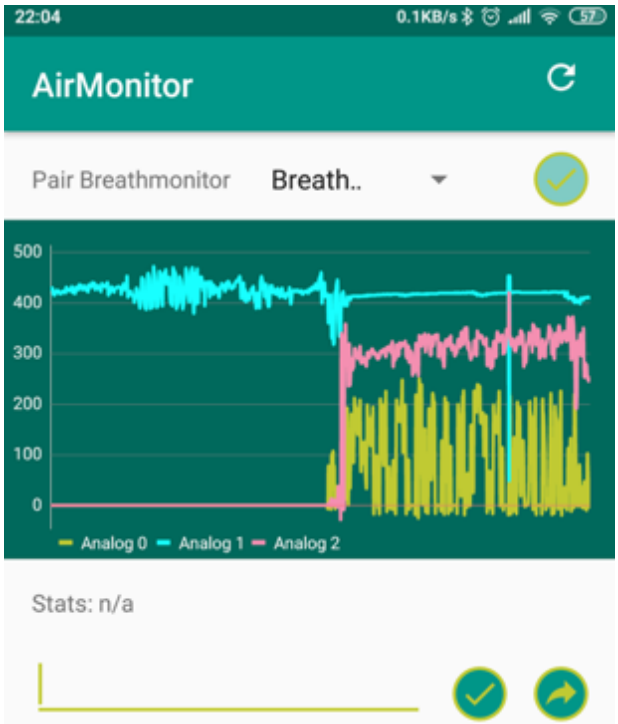


Рисунок 27 Розроблений Android додаток для збору даних

Щоб отримати більш збалансовані дані, було зібрано тестову групу з 8 користувачів (по 4 людини кожної статі), 4 з них були 25-річними, 2 - 40-ка річними і 2 - 55 річними. У всіх користувачів, включених в тестову групу, був середній рівень фізичної активності. Виміри проводились за допомогою смартфона з додатком і приладом, прикріпленим до грудної клітки (Рис. 4.10). Кожна дія записувалась протягом 5 хвилин (для запису після віджимань записувалось значення протягом 30 секунд, а потім повторювалася серія з 10 віджимань). Таким чином, повний набір даних включає в себе 432 000 значень.

Після сеансу запису дані зберігалися на мобільному телефоні, а потім розмічались за допомогою додавання стовпчика з маркером активності. Отриманий набір даних поділявся на частини для тестування і навчання. Оскільки вимірювання велись з узагальненими даними, що описують діяльність людини, то для перевірки працездатності та точності моделі, дані, зібрані в одного з учасників зберігалися окремо в якості тестового набору даних. Оскільки у різних людей різний об'єм легень, рух грудної клітини та інші фізіологічні параметри, то, якщо методика справедлива для людини, що не була додана до тестової вибірки, то її можна використовувати і для інших людей що не входили в тестовий набір даних.

4.4. Обробка даних та визначення типу дихання

Акселерометр та сенсор тиску виробляють одномірні сигнали, на відміну від двовимірних зображень. Отже, в рамках обробки даних перед класифікацією, сигнал було перетворено в 2d-зображення і отримані результати порівняно з традиційним підходом з використання 1d сигналу.

Перетворення 1d-сигналу в 2d-зображення можна виконати кількома способами:

- Використання технології відображення активності
- Використання нормалізації і перетворення в матрицю

Ці методи можуть використовуватися в різних ситуаціях для різних сигналів.

Використовуючи перетворення в матрицю, зразки сигналу всередині вікна перетворюються у зображення сірого кольору зі шкалою 0-255. У цьому зображені темніший колір позначає велику амплітуду в вихідному сигналі, а координати пікселя - (i, j) матриці M × N для кожного n-го зразка сигналу, де

$$i = \frac{N}{M} \text{ и } j = \frac{N}{M}.$$

На Рис. 4.9 ілюструється цей підхід.

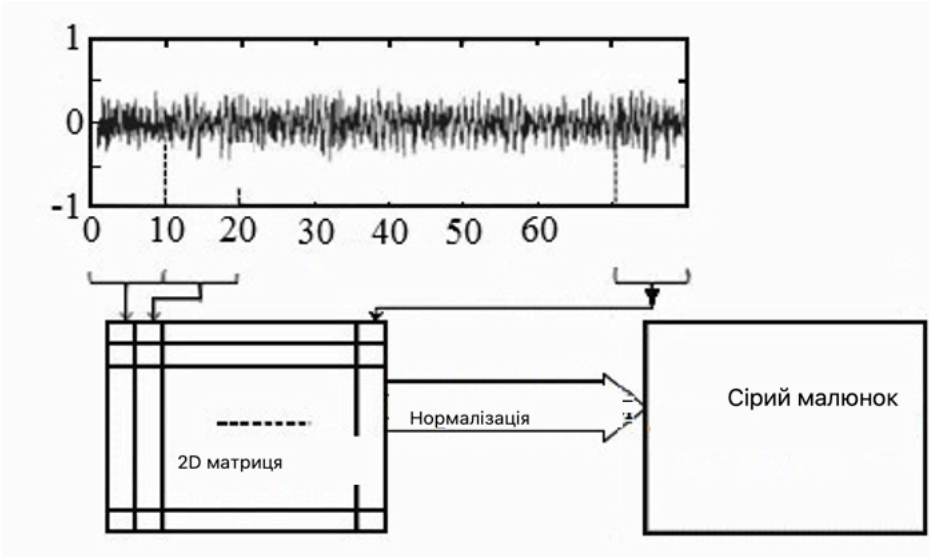


Рисунок 28 Перетворення сигналу в матрицю

Цей підхід простий і зручний в роботі, так як він не вимагає великої вичислительной потужності, але, в той же час, має дуже великий недолік: він дуже сприйнятливий до шуму і аномалій, тому він не може використовуватися для даних з сенсорів.

Як показано в роботі [161], альтернативою може бути підхід, коли сигнали складаються разом послідовно по рядках (stacking) у вигляді зображения, що дозволяє кожній послідовності сигналів корелювати з іншими послідовностями. Потім проводиться 2d дискретне перетворень-тання Фур'є (DFT), а його величина являє собою зображення, яке використовується в якості вхідного сигналу для CNN. Цей підхід модіфікується включенням процедури нормалізації сигналів перед стекінгом в зображення.

Рисунок 29 процес збору тренувального набору даних

Дані, зібрані з сенсора, мають наступну структуру: часову мітку (у форматі Unix з мілісекундами), прискорення по осі X, прискорення по осі Y, прискорення по осі Z, значення сенсора тиску. Наприклад:

1486291984848|1.61407470703125|-9.95855712890625| 2.83917236328125.

Оскільки сам акселерометр вимірює гравітаційне прискорення (g) і лінійне прискорення (a), ми бачимо, що в міру переміщення людей в різних напрямках акселерометр обертається, і кожна вісь вимірює прискорення вздовж різних осей (g або a). Щоб безперервно вимірювати рух на потрібній осі і зменшувати шум (відключити нерелевантні вимірювання), необхідно нормалізувати значення вимірювань.

Кут θ_t показує, як обертання осі від часу t-1 до t:

$$\theta_t = cos^{-1}(a_t \times a_{t-1})$$

де a - вектор значень за час. Щоб зменшити вплив шуму на результати, кожне спостереження нормалізується на кут обертання θt, в момент часу t. Таким чином, нормалізований сигнал буде виглядати так:

$$a_{xt} = a_x \times \theta_t$$

$$a_{yt} = a_y \times \theta_t$$

$$a_{zt} = a_z \times \theta_t$$

Оскільки використовується метод ковзного вікна, то, щоб ще зменшити шум, застосовується функцію H(n) вікна Хеммінга на нормалізованому сигналі.

В результаті алгоритм попередньої обробки необробленого вхідного сигналу виглядає наступним чином:

1. Нормалізувати вхідний сигнал щодо осі обертання, як опи-сано вище.
2. Застосувати функцію вікна Хеммінга на вже нормалізованому си-ле.
3. Перетворити сигнал з нашого вікна, за допомогою алгоритму з [13].
4. Застосуйте DFT на отриманому зображенні.

Після виконання цього підходу на даних з сенсорів результати передаються в CNN. Процес обробки сигналу показаний на Рис. 4.11

Для проведення експериментів використовувалася стратегія ковзного вікна для обробки сигналів як стосовно до вихідного сигналу, так і для випадку генерації зображень з вхідного сигналу. Основна ідея ковзного вікна полягає в тому, щоб розділити сигнал часових рядів на короткі фрагменти.

Коли дані знімаються з частотою 30 спостережень в секунду, то розмір ковзаючого вікна, рівний 90 вибірках, відповідає 3 секундам спостережень, а розмір кроку вікна, рівний 32 вибірках, становить 1,06 секунди. Більший чи менший розмір кроку може використовуватися або для збільшення точності (менше), або для зменшення необхідної обчислювальної потужності (більше), споживаної енергії, що є критичним параметром. Тому повинен бути знайдений компроміс між точністю і кількістю обчислень.

Так як основна мета дослідження - виявити тип дихання в реальному часі з мінімальною затримкою, вища обчислювальна вартість може бути проблемою, що викликає великі затримки (особливо на не дуже потужних пристроях).

Для першого експерименту з вхідними даними у вигляді нормалізуватиного сигналу генеруються сегменти вікон і додаються в кожен компонент

сигналу у вигляді третього виміру, щоб вхідний вектор для CNN був виду: [повні сегменти, вхідний сегмент, вхід канал].

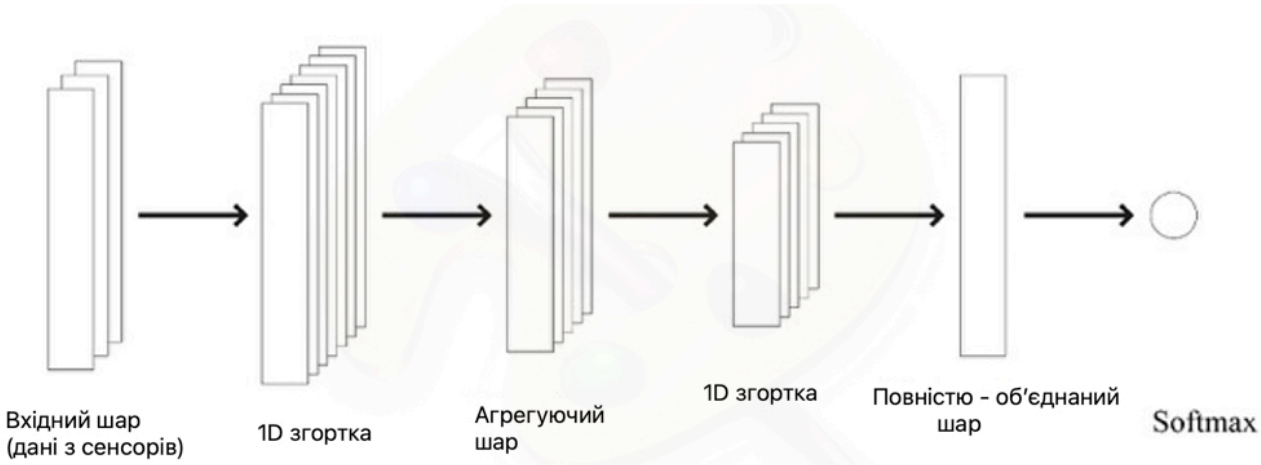
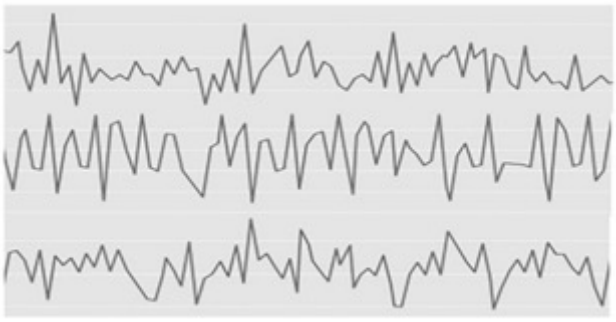


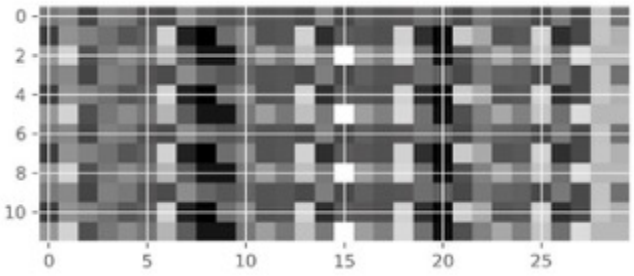
Рисунок 30 Архітектура CNN для першого експерименту [63]

Оскільки сигнал є одновимірним і для нього передбачена 1d-згортка, то необхідно змінити сформовані вікна, які будуть нормалізовані до висоти 1.

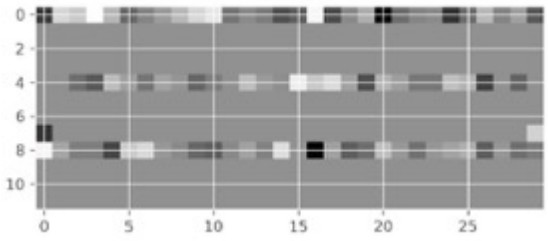
Для другого експерименту з стекінгом і обробкою сигналів як графічного зображення вводяться зміни. Оскільки є 3 послідовності вхідних сигналів, то відповідно кожен сигнал повторюється чотири рази. В результаті чого розмір зображення вхідного сигналу дорівнює 12x32, де параметр 32 визначається вибраним вікна [63] (Рис. 4.12).



а



б



в

Рисунок 31 Обробка сигналу акселерометра: а - вхідний сигнал; б - сигнальний рисунок; в - спектр амплітуди

Всі шари CNN можна згрупувати, як описано нижче:

Перший шар згортки має розмір фільтра і глибину 60 (кількість каналів отримують як вихід з шару згортки). Розмір фільтра шару об'єднання встановлюється рівним 20 з кроком 2. Потім шар згортки бере вхідний шар з максимальним рівнем об'єднання, застосовуючи фільтр розміром 6, і буде мати десяту частину глибини на рівні максимального об'єднання. Після цього вихід згладжується для вектора входу повністю підключеного шару [63].

У повністю підключеному шарі (це може бути визначено конфігурацією) розміщені тисячі нейронів. У цьому шарі використовується функція гіперболічні тангенси tanh для нелінійності. Шар Softmax визначається для виведення ймовірностей міток класу. Функція негативного логарифмічна-го правдоподібності (negative log likelihood) [162] мінімізується за допомогою

стохастичного оптимізатора спуску градієнта (SGD).

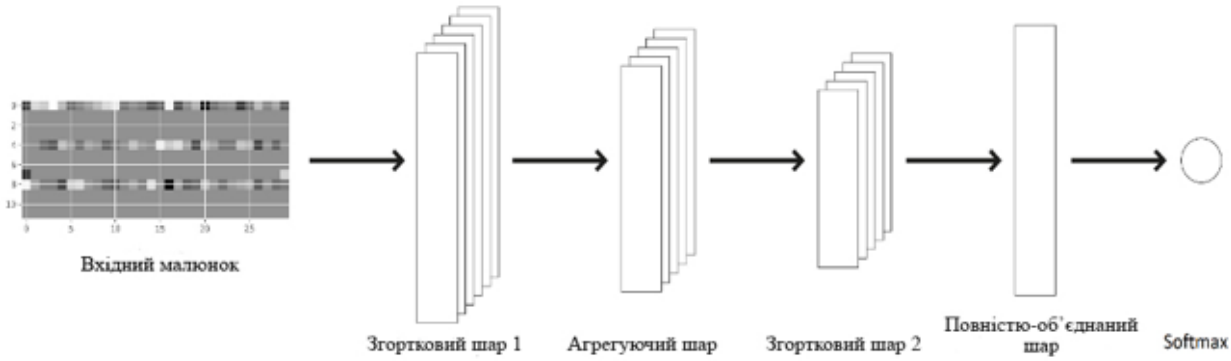


Рисунок 32 CNN для виявлення активності людини

Використання вихідного сигналу (з деякою попередньої обробкою) в архітектурі CNN є нормальною ситуацією для додатків глибинного навчання в предметної області комп'ютерного зору. Однак CNN, як правило, не настільки ефективні з 1-мірним сигналом і для про-ведення другого експерименту здійснюється перетворення сигналу в 2D-зображення.

Для експерименту з стекінгом вхідного сигналу, структура запропонованої нейронної мережі для класифікації дихання зображена на Рис. 4.13:

Модель складається з одного шару згортки, за яким слідує шар максимального об'єднання (pooling layer), і ще один шар згортки. Після цього модель містить повністю з’єднаний шар (fully-connected layer), який підключений до шару Softmax.

CNN навчаються з використанням ітеративної оптимізації за допомогою алгоритму зворотного поширення (backpropagation). Найбільш широко використовуваним методом оптимізації є стохастичний градієнтний спуск (SGD). В експерименті використовувався оптимізатор модифікований SGD, так як він має найнижчу помилку навчання і динамічну швидкість навчання. Функція вартості навчання CNN-архітектури - Softmax з L2-регуляризацією. В мережі використана випрямлена лінійна одиниця (ReLU) як функція активації (activation function).

Гіперпараметрами CNN є:

- кількість і типи шарів

- розмір фільтрів для згортки і крок згортки для кожного сверточного шару
- розмір області об'єднання
- крок об'єднання для кожного шару об'єднання
- кількість одиниць для кожного повністю підключеного шару

Процес навчання мережі може бути описаним наступним чином:

- Перший шар згортки має розмір фільтра і глибину 60 (кількість каналів отримують як вихід з шару згортки). Розмір фільтра шару об'єднання встановлюється рівним 20 з кроком 2.
- Шар згортки бере вхідний шар з максимальним рівнем об'єднання, застосовуючи фільтр розміром 6
- Після цього вихід згладжується для вектора входу повністю підключеного шару (fully connected layer).
- В повністю підключеному шарі (це може бути визначено конфігурацією) розміщені 1000 нейронів. У цьому шарі використовується функція гіперболічного тангенса tanh для нелінійності.
- Шар Softmax використовується для виведення ймовірностей міток класу.

Функція негативного логарифмічного правдоподібності (negative log likelihood) мінімізується за допомогою стохастичного оптимізатора спуску градієнта (SGD).Таким чином, класифікацію можна представити наступною функцією:

$$y_i = \left(1 + exp\left(b_j + \sum_{i}^{j} k_{ij} \times x_i \right) \right)^{-1}$$

де k_{ij} - згорткове ядро на i-тому вхідному відображенні x_i для генерації j-го вихідного відображення y_j, b_j.

Шар об'єднання (pooling layer) організований як метод підвибірки. Вихід y_i обчислюється шляхом прийняття середніх значень областей різних класів x_i з фільтром m x m

$$y_i(r, c) = \frac{\sum_{p=1}^{m} \sum_{q=1}^{m} x_i(r \times m + p, c \times m + q)}{m^2},$$

де *r, c* - координати пікселя y_i.

Після повністю підключеного шару ми отримуємо 1D-вектор f на виході. Функція Softmax застосовується для розподілення ймовірності по всіх класах (в даному випадку, оскільки є 6 різних типів дихання, отримується можливість для кожного класу):

$$p(s) = \frac{g_s}{\sum_{j=1}^{N_a} g_j}, \text{ де } g_j = max\left(0, \sum_i^j f_i \times w_{ij} + h_j\right),$$

де *w* і *h* - коефіцієнти функції softmax, а *s* - один з класів, що передбачаються.

4.5. Оцінка і порівняння точності виділення шаблонів дихання

Основною проблемою для оцінки результатів була проблема дисбалансу класів. Набагато легше реєструвати низьку фізичну активність для багатьох, особливо старших, людей (які є основною цільовою групою для якої розробляється рішення). В результаті чого було отримано набагато більше даних, записаних для спокійного дихання, ніж для інших типів. Щоб зменшити вплив незбалансованих класів, можна використовувати різні збалансовані групи даних для кожної епохи в оптіміза-торі YellowFin [162].

Для оцінки результатів побудованої моделі обчислювалися такі показники:

- Точність, відгук (precision, recall),
- F1,
- Втрата навчання (training loss),
- Точність навчання (accuracy).

Точність тестування перевірялася, використовуючи записи людини, яка не була додана в навчальний набір даних, що допомогло уникнути перенавчання на конкретних людях.

Оскільки в роботі класифікуються 6 різних класів (шаблонів дихання) з бінарними метриками [163], то необхідно представляти цю класифікацію як 6 різних бінарних класифікацій, де кожен раз один клас вважається позитивним, а всі інші класи позначені як негативні (класифікація один-проти всіх). Цьому

сприяє побудова матриці негараздів (confusion matrix), наведеної нижче на Табл. 3 для кращого розуміння продуктивності побудованої моделі.

З табл. 4 видно, що після 40 епох навчання побудована модель явно почала перенавчатися, в результаті чого досягається рівень 84% точності моделі. Те ж саме можна побачити в табл. 5, де для попередньо обробленого сиг-налу показана точність 88%, що, безсумнівно, краще, ніж в попередньому випадку, але все ж далеко від досконалості.

Таблиця 4

Метрики з використанням нормалізованого сигналу

Кількість епох	F1	Recall	Precision	Training Accuracy	Training Loss	Testing Accuracy
Epoch 0	0.615	0.67	0.6	0.67	7.82	0.45
Epoch 10	0.839	0.881	0.825	0.881	3.63	0.765
Epoch 20	0.942	0.947	0.941	0.947	2.653	0.818
Epoch 30	0.969	0.971	0.97	0.971	2.179	0.834
Epoch 40	0.976	0.977	0.977	0.977	1.81	0.843
Epoch 50	0.98	0.98	0.98	0.98	1.709	0.836

Таблиця 5

Метрики з використанням перетворення сигналу в малюнок

Кількіст епох	F1	Recall	Precision	Training ccuracy	Training oss	Testing ccuracy
Epoch 0	0.515	0.57	0.5	0.57	5.73	0.37
Epoch 10	0.727	0.732	0.749	0.76	4.02	0.65
Epoch 20	0.919	0.918	0.925	0.95	2.756	0.729
Epoch 30	0.959	0.951	0.939	0.961	2.217	0.8
Epoch 40	0.973	0.97	0.981	0.98	1.756	0.88
Epoch 50	0.983	0.983	0.983	0.983	1.69	0.873

Таблиця 6

Матриця негараздів CNN для необробленого сигналу

Тип дихання	**Спокійне**	**Після віджимання**	**Глибоке**	**Швидке**	**Під час бігу**	**З затримкою**
З затримкою	0.97	0.028	0	0	0	0
Під час бігу	0.026	0.64	0.33	0	0	0
Швидке	0	0.2	0.75	0	0	0.055
Глибоке	0	0	0	0.65	0.33	0.014
Після віджимання	0	0	0	0.24	0.76	0
Спокійне	0	0.051	0.029	0	0	0.92

Таблиця 7

Матриця негараздів CNN для сигналу, який можна конвертувати в малюнок

Тип дихання	**Спокійне**	**Після віджимання**	**Глибоке**	**Швидке**	**Під час бігу**	**З затримкою**
З затримкою	0.95	0.041	0.014	0	0	0
Під час бігу	0.021	0.73	0.23	0.021	0	0
Швидке	0	0.19	0.72	0.017	0.069	0
Глибоке	0	0.015	0.015	0.71	0.26	0
Після віджимання	0	0.043	0	0.19	0.77	0
Спокійне	0	0	0.029	0	0	0.97

Як видно з матриць негараздів, наведених в табл. 6, 7, запропонований підхід досить точно може визначити спокійне дихання і дихання з затримкою, в той час як при спробі класифікувати інші типи фіксується багато помилкових спрацьовувань. Особливо багато помилкової класифікації між бігом і швидким диханням, а також між віджиманнями і глибоким диханням. Це можна пояснити

тим, що сигнали від дихання з затримкою та спокійним диханням з меншою ймовірністю супроводжуються іншими діями, пов'язаними з рухом грудної клітки, і тому більш унікальні. У той час як при швидкому диханні і бігу (а також віджиманні і глибокому диханні), рухи грудної клітини більш ідентичні і, крім того, вхідний сигнал забруднений величезною кількістю даних фізичних активностей, через що дихальне рух грудної клітини розпізнається з похибкою.

Висновки до розділу 4

1. Досліджено побудову лямбда-архітектури з врахуванням прикінцевих обчислень для збору даних з сенсорів, та розроблено систему постійного моніторингу дихання пацієнта на основі аналізу руху його грудної клітки та діафрагми з допомогою акселерометра, гіроскопа та сенсорів механічної деформації.
2. Продемонстровано ефективність підходу підвищення ефективності застосування мережі CNN використовуючи попередню обробку вхідних сигналів за допомогою стекінга в умовах наявності шумів від інших джерел та рухів людини.
3. Розроблена система сенсорів, хмарний та мобільний додатки використовувалася для збору та збереження даних під час роботи над публікацією [1], а також була представлена та стала переможцем конкурсу Sikorsky challenge-2018 у номінації «Краще технологічне рішення».

Висновки

У дисертаційній роботі розв’язано актуальну науково-практичну задачу розробки методики розпізнавання людських дій за допомогою методів машинного навчання. Запропоновані методи збору даних з сенсорів, збереження їх в хмарі та класифікація за допомогою методів машинного навчання не просто дозволяє розрізняти різні рухи людини, а також може використовуватися в медичних цілях в якості постійного моніторингу за станом людини. Це в свою чергу дозволяє отримати швидкий сигнал про зміну стану людини (наприклад приступ апное) і застосувати відповідні заходи.

В результаті дослідження отримані наступні теоретичні і практичні результати, покликані допомогти в розв’язку наведеної вище задачі:

1. Аналіз існуючих систем розпізнавання дій людини показав, що певні недоліки і незручності використання портативних натільних датчиків низького рівня можна компенсувати за рахунок методів машинного навчання і тим забезпечити багату контекстуальну інформацію в реальному застосуванні. Тому на відміну від існуючих прототипів в роботі наголос зроблено на дослідження властивостей таких систем при використанні новітніх згорткових нейронних мереж (CNN).
2. Вперше запропоновано для підвищення ефективності застосування мережі CNN перетворювати одновимірні сигнали (1d) акселерометра в двовимірні (2d) графічні зображення , які оброблюються за допомогою CNN із декількома обробними шарами, завдяки чому точність визначення типу дії людини в різних ситуаціях для різних фізичних станів зростає в порівнянні з випадком, коли двовимірні перетворення сигналів акселерометра не вживаються.
3. Вперше для оброблення сигналів датчика-акселерометра в умовах наявності перешкоджаючих сигналів (шумів) від інших джерел та інструментальних похибок пристрою запропоновано перетворити сигнал з допомогою вікна Хеммінга, складаючи сигнали разом послідовно по

рядках (stacking) у вигляді зображення, що дозволяє кожній послідовності сигналів корелювати з іншими послідовностями.

4. Розроблений набір даних для навчання і тестування системи, для збору якого були залучені люди різного віку і з різною фізичною активністю, що дозволяє частково зменшити вплив індивідуальності рухів кожної людини на навчання моделі.
5. Вперше висунута і перевірена гіпотеза про можливість застосування систем розпізнавання дій людини для моніторингу руху грудної клітини, через який визначається частота і глибина вдихів при диханні, притаманні певним хворобам, включаючи сонне апное (зупинка дихання під час сну).

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ:

[1] O. D. Lara і M. A. Labrador, «A Survey on Human Activity Recognition using Wearable Sensors», *IEEE Commun. Surv. Tutor.*, вип. 15, вип. 3, с. 1192–1209, Third 2013, doi: 10.1109/SURV.2012.110112.00192.

[2] C. Jobanputra, J. Bavishi, і N. Doshi, «Human Activity Recognition: A Survey», *Procedia Comput. Sci.*, вип. 155, с. 698–703, Січ 2019, doi: 10.1016/j.procs.2019.08.100.

[3] A. Godfrey, R. Conway, D. Meagher, і G. OLaighin, «Direct measurement of human movement by accelerometry», *Med. Eng. Phys.*, вип. 30, вип. 10, с. 1364–1386, Груд 2008, doi: 10.1016/j.medengphy.2008.09.005.

[4] A. Ignatov, «Real-time human activity recognition from accelerometer data using Convolutional Neural Networks», *Appl. Soft Comput.*, вип. 62, с. 915–922, Січ 2018, doi: 10.1016/j.asoc.2017.09.027.

[5] C. A. Ronao і S.-B. Cho, «Human activity recognition with smartphone sensors using deep learning neural networks», *Expert Syst. Appl.*, вип. 59, с. 235–244, Жов 2016, doi: 10.1016/j.eswa.2016.04.032.

[6] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, і Y. Amirat, «Physical Human Activity Recognition Using Wearable Sensors», *Sensors*, вип. 15, вип. 12, Art. вип. 12, Груд 2015, doi: 10.3390/s151229858.

[7] Z. Wang, C. Zhao, і S. Qiu, «A system of human vital signs monitoring and activity recognition based on body sensor network», *Sens. Rev.*, вип. 34, вип. 1, с. 42–50, Січ 2014, doi: 10.1108/SR-12-2012-735.

[8] «MPU-9250 | TDK». https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/ (дата звернення Бер 24, 2021).

[9] W. Zijlstra і A. L. Hof, «Assessment of spatio-temporal gait parameters from trunk accelerations during human walking», *Gait Posture*, вип. 18, вип. 2, с. 1–10, Жов 2003, doi: 10.1016/s0966-6362(02)00190-x.

[10] T. He, J. A. Stankovic, T. F. Abdelzaher, і C. Lu, «A spatiotemporal communication protocol for wireless sensor networks», *IEEE Trans. Parallel Distrib. Syst.*, вип. 16, вип. 10, с. 995–1006, Жов 2005, doi: 10.1109/TPDS.2005.116.

[11] J. G. Proakis і D. G. Manolakis, *Digital signal processing : principles, algorithms, and applications*. Upper Saddle River, N.J. : Prentice Hall, 1996.

[12] «Outlier Detection & Analysis: The Different Types of Outliers», *Anodot*, Лип 31, 2019. https://www.anodot.com/blog/quick-guide-different-types-outliers/ (дата звернення Бер 24, 2021).

[13] S. Salvador, P. Chan, і J. Brodie, «Learning States and Rules for Time Series Anomaly Detections», с. 6.

[14] K. K. L. B. Adikaram, M. A. Hussein, M. Effenberger, і T. Becker, «Data Transformation Technique to Improve the Outlier Detection Power of Grubbs' Test for Data Expected to Follow Linear Relation», *J. Appl. Math.*, вип. 2015, с. e708948, Січ 2015, doi: 10.1155/2015/708948.

[15] *8.9 Seasonal ARIMA models | Forecasting: Principles and Practice (2nd ed)*. .

[16] P. Tiunov, «Time Series Anomaly Detection Algorithms», *Medium*, Бер 07, 2019. https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2 (дата звернення Бер 24, 2021).

[17] A. L. N. Reddy і P. Banerjee, «Algorithm-based fault detection for signal processing applications», *IEEE Trans. Comput.*, вип. 39, вип. 10, с. 1304–1308, Жов 1990, doi: 10.1109/12.59860.

[18] P. Catalfamo, S. Ghoussayni, і D. Ewins, «Gait Event Detection on Level Ground and Incline Walking Using a Rate Gyroscope», *Sensors*, вип. 10, вип. 6, с. 5683–5702, Чер 2010, doi: 10.3390/s100605683.

[19] E. Sejdić, I. Djurović, і J. Jiang, «Time–frequency feature representation using energy concentration: An overview of recent advances», *Digit. Signal Process.*, вип. 19, вип. 1, с. 153–183, Січ 2009, doi: 10.1016/j.dsp.2007.12.004.

[20] V. Britanak, P. C. Yip, і K. R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Elsevier, 2010.

[21] D. Hsu, S. M. Kakade, і T. Zhang, «A Spectral Algorithm for Learning Hidden Markov Models», *ArXiv08114413 Cs*, Лип 2012, Дата звернення: Бер 25, 2021. [Online]. Доступний у: http://arxiv.org/abs/0811.4413.

[22] Ş. Büyüköztürk і Ö. Çokluk-Bökeoğlu, «Discriminant Function Analysis : Concept and Application Ş ener Büyüköztürk», 2008. /paper/Discriminant-Function-Analysis-%3A-Concept-and-%C5%9E-ener-B%C3%BCy%C3%BCk%C3%B6zt%C3%BCrk-%C3%87okluk-B%C3%B6keo%C4%9Flu/76b1c725b926d91db6eb52e83ab1596e6eb115ca (дата звернення Бер 25, 2021).

[23] T. Huynh і B. Schiele, «Analyzing features for activity recognition», в *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, New York, NY, USA, Жов 2005, с. 159–163, doi: 10.1145/1107548.1107591.

[24] Z. Sheng, C. Hailong, J. Chuan, і Z. Shaojun, «An adaptive time window method for human activity recognition», в *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Трав 2015, с. 1188–1192, doi: 10.1109/CCECE.2015.7129445.

[25] E. Brophy, W. Muehlhausen, A. F. Smeaton, і T. E. Ward, «Optimised Convolutional Neural Networks for Heart Rate Estimation and Human Activity Recognition in Wrist Worn Sensing Applications», *ArXiv200400505 Cs Eess Stat*, Бер 2020, Дата звернення: Бер 25, 2021. [Online]. Доступний у: http://arxiv.org/abs/2004.00505.

[26] A. Dehghani, O. Sarbishei, T. Glatard, і E. Shihab, «A Quantitative Comparison of Overlapping and Non-Overlapping Sliding Windows for Human Activity Recognition Using Inertial Sensors», *Sensors*, вип. 19, вип. 22, Art. вип. 22, Січ 2019, doi: 10.3390/s19225026.

[27] G. I. Webb, «Naïve Bayes», в *Encyclopedia of Machine Learning*, C. Sammut і G. I. Webb, Ред. Boston, MA: Springer US, 2010, с. 713–714.

[28] C. Cortes і V. Vapnik, «Support-vector networks», *Mach. Learn.*, вип. 20, вип. 3, с. 273–297, Вер 1995, doi: 10.1007/BF00994018.

[29] E. P. Ijjina і K. M. Chalavadi, «Human action recognition using genetic algorithms and convolutional neural networks», *Pattern Recognit.*, вип. 59, с. 199–212, Лис 2016, doi: 10.1016/j.patcog.2016.01.012.

[30] «Large Margin Classification Using the Perceptron Algorithm | SpringerLink». https://link.springer.com/article/10.1023/A:1007662407062 (дата звернення Бер 24, 2021).

[31] Z. S. Abdallah, M. M. Gaber, B. Srinivasan, і S. Krishnaswamy, «Adaptive mobile activity recognition system with evolving data streams», *Neurocomputing*, вип. 150, с. 304–317, Лют 2015, doi: 10.1016/j.neucom.2014.09.074.

[32] D. Graupe, «Applications of signal and image processing to medicine», Січ 1988, doi: 10.1109/ISCAS.1988.15362.

[33] «Using the CNN Architecture in Image Processing», *Open Data Science - Your News Source for AI, Machine Learning & more*, Січ 09, 2020. https://opendatascience.com/using-the-cnn-architecture-in-image-processing/ (дата звернення Бер 25, 2021).

[34] M. Z. Amin і N. Nadeem, «Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System», *ArXiv180902479 Cs*, Жов 2019, Дата звернення: Бер 25, 2021. [Online]. Доступний у: http://arxiv.org/abs/1809.02479.

[35] P. Radhakrishnan, «Introduction to Recurrent Neural Network», *Medium*, Січ 13, 2018. https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3 (дата звернення Бер 24, 2021).

[36] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, і L.-S. Lee, «Audio Word2Vec: Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Autoencoder», *ArXiv160300982 Cs*, Чер 2016, Дата звернення: Бер 25, 2021. [Online]. Доступний у: http://arxiv.org/abs/1603.00982.

[37] D. Wang, J. Gong, і Y. Song, «W-RNN: News text classification based on a Weighted RNN», *ArXiv190913077 Cs Stat*, Вер 2019, Дата звернення: Бер 25, 2021. [Online]. Доступний у: http://arxiv.org/abs/1909.13077.

[38] S. Hochreiter і J. Schmidhuber, «Long Short-term Memory», *Neural Comput.*, вип. 9, с. 1735–80, Груд 1997, doi: 10.1162/neco.1997.9.8.1735.

[39] 2018 at 7:29am Posted by William Vorhies on May 1 і V. Blog, «Temporal Convolutional Nets (TCNs) Take Over from RNNs for NLP Predictions». https://www.datasciencecentral.com/profiles/blogs/temporal-convolutional-nets-tcns-take-over-from-rnns-for-nlp-pred (дата звернення Бер 25, 2021).

[40] R. Roy, «TEMPORAL CONVOLUTIONAL NETWORKS», *Medium*, Лют 04, 2019. https://medium.com/@raushan2807/temporal-convolutional-networks-bfea16e6d7d2 (дата звернення Бер 25, 2021).

[41] «Transformers — transformers 4.4.2 documentation». https://hugging-face.co/transformers/ (дата звернення Бер 25, 2021).

[42] Y. Sasaki, «The truth of the F-measure», с. 5.

[43] O. Incel, M. Kose, і C. Ersoy, «A Review and Taxonomy of Activity Recognition on Mobile Phones», *BioNanoScience*, вип. 3, Чер 2013, doi: 10.1007/s12668-013-0088-3.

[44] U. Maurer, A. Rowe, A. Smailagic, і D. P. Siewiorek, «eWatch: a wearable sensor and notification platform», в *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, Квіт 2006, с. 4 pp. – 145, doi: 10.1109/BSN.2006.24.

[45] M. A. Labrador і O. D. L. Yejas, *Human Activity Recognition: Using Wearable Sensors and Smartphones*. CRC Press, 2013.

[46] «BioHarness | BIOPAC», *BIOPAC Systems, Inc.* https://www.bi-opac.com/product-category/research/telemetry-and-data-logging/bioharness/ (дата звернення Бер 25, 2021).

[47] D. Riboni і C. Bettini, «COSAR: hybrid reasoning for context-aware activity recognition», *Pers. Ubiquitous Comput.*, вип. 15, вип. 3, с. 271–289, Бер 2011, doi: 10.1007/s00779-010-0331-7.

[48] T. Kao, C. Lin, і J. Wang, «Development of a portable activity detector for daily activity recognition», в *2009 IEEE International Symposium on Industrial Electronics*, Лип 2009, с. 115–120, doi: 10.1109/ISIE.2009.5222001.

[49] G. E. Moore, «Cramming more components onto integrated circuits», вип. 38, вип. 8, с. 4, 1965.

[50] S. Ullah *et al.*, «A Comprehensive Survey of Wireless Body Area Networks», *J. Med. Syst.*, вип. 36, вип. 3, с. 1065–1094, Чер 2012, doi: 10.1007/s10916-010-9571-3.

[51] *Small Machines, Large Opportunities: A Report on the Emerging Field of Microdynamics : Report of the Workshop on Microelectromechanical Systems Research ; Sponsored by the National Science Foundation*. AT & T Bell Laboratories, 1988.

[52] F. A. Storm, C. J. Buckley, і C. Mazzà, «Gait event detection in laboratory and real life settings: Accuracy of ankle and waist sensor based methods», *Gait Posture*, вип. 50, с. 42–46, Жов 2016, doi: 10.1016/j.gaitpost.2016.08.012.

[53] A. M. J. Sarkar, «An Intelligent Tool for Activity Data Collection», *Sensors*, вип. 11, вип. 4, Art. вип. 4, Квіт 2011, doi: 10.3390/s110403988.

[54] Huikai Xie і G. K. Fedder, «Fabrication, characterization, and analysis of a DRIE CMOS-MEMS gyroscope», *IEEE Sens. J.*, вип. 3, вип. 5, с. 622–631, Жов 2003, doi: 10.1109/JSEN.2003.817901.

[55] M. Elwenspoek і R. Wiegerink, *Mechanical Microsensors*. Berlin Heidelberg: Springer-Verlag, 2001.

[56] Z. Diao, H. Quan, L. Lan, і Y. Han, «Analysis and compensation of MEMS gyroscope drift», в *2013 Seventh International Conference on Sensing Technology (ICST)*, Груд 2013, с. 592–596, doi: 10.1109/ICSensT.2013.6727722.

[57] M. Marinov і Z. Petrov, «ALLAN VARIANCE ANALYSIS ON ERROR CHARACTERS OF LOW-COST MEMS ACCELEROMETER MMA8451Q», Трав 2016.

[58] J. Shiau, C. Huang, і M. Chang, «Noise Characteristics of MEMS Gyro’s Null Drift and Temperature Compensation», 2012, doi: 10.6180/JASE.2012.15.3.04.

[59] L. Keselbrener, M. Keselbrener, і S. Akselrod, «Nonlinear high pass filter for R-wave detection in ECG signal», *Med. Eng. Phys.*, вип. 19, вип. 5, с. 481–484, Чер 1997, doi: 10.1016/S1350-4533(97)00013-1.

[60] «fsrguide.pdf». Дата звернення: Бер 24, 2021. [Online]. Доступний у: https://www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf.

[61] K. Tong і M. H. Granat, «A practical gait analysis system using gyroscopes», *Med. Eng. Phys.*, вип. 21, вип. 2, с. 87–94, Бер 1999, doi: 10.1016/s1350-4533(99)00030-2.

[62] A. Roshan Fekr, M. Janidarmian, K. Radecka, і Z. Zilic, «Movement analysis of the chest compartments and a real-time quality feedback during breathing therapy», *Netw. Model. Anal. Health Inform. Bioinforma.*, вип. 4, Груд 2015, doi: 10.1007/s13721-015-0093-2.

[63] A. Petrenko, R. Kyslyi, і I. Pysmennyi, «Detection of human respiration patterns using deep convolution neural networks», *East.-Eur. J. Enterp. Technol.*, вип. 4, вип. 9 (94), Art. вип. 9 (94), Сер 2018, doi: 10.15587/1729-4061.2018.139997.

[64] J. Vertens *et al.*, «Measuring Respiration and Heart Rate using Two Acceleration Sensors on a Fully Embedded Platform», Лис 2015, doi: 10.5220/0005604000150023.

[65] «Respiration rate and volume measurements using wearable strain sensors | npj Digital Medicine». https://www.nature.com/articles/s41746-019-0083-3 (дата звернення Бер 24, 2021).

[66] «The fast Fourier transform and its applications | Guide books». https://dl.acm.org/doi/abs/10.5555/47314 (дата звернення Бер 25, 2021).

[67] «Narrow-Bandpass Waveguide Filters». https://ieeexplore.ieee.org/abstract/document/1127732/ (дата звернення Бер 25, 2021).

[68] I. W. Selesnick і C. S. Burrus, «Generalized digital Butterworth filter design», *IEEE Trans. Signal Process.*, вип. 46, вип. 6, с. 1688–1694, Чер 1998, doi: 10.1109/78.678493.

[69] G.-Z. Liu, Y.-W. Guo, Q.-S. Zhu, B.-Y. Huang, і L. Wang, «Estimation of respiration rate from three-dimensional acceleration data based on body sensor network», *Telemed. J. E-Health Off. J. Am. Telemed. Assoc.*, вип. 17, вип. 9, с. 705–711, Лис 2011, doi: 10.1089/tmj.2011.0022.

[70] A. Bates, M. J. Ling, J. Mann, і D. K. Arvind, «Respiratory Rate and Flow Waveform Estimation from Tri-axial Accelerometer Data», в *2010 International Conference on Body Sensor Networks*, Чер 2010, с. 144–150, doi: 10.1109/BSN.2010.50.

[71] «algorithm - Peak signal detection in realtime timeseries data», *Stack Overflow*. https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data (дата звернення Бер 25, 2021).

[72] «Structural break», *Wikipedia*. Бер 18, 2021, Дата звернення: Бер 25, 2021. [Online]. Доступний у: https://en.wikipedia.org/w/index.php?title=Structural_break&oldid=1012840805.

[73] B. M. R. Lima *et al.*, «Heart Rate Detection Using a Miniaturized Multimodal Tactile Sensor», в *2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Чер 2019, с. 1–6, doi: 10.1109/MeMeA.2019.8802209.

[74] «Deep Learning». https://www.deeplearningbook.org/ (дата звернення Бер 26, 2021).

[75] M. Sakurada і T. Yairi, «Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction», в *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, New York, NY, USA, Груд 2014, с. 4–11, doi: 10.1145/2689746.2689747.

[76] K. Cho *et al.*, «Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation», *ArXiv14061078 Cs Stat*, Вер 2014, Дата звернення: Квіт 08, 2021. [Online]. Доступний у: http://arxiv.org/abs/1406.1078.

[77] G. E. Hinton і R. R. Salakhutdinov, «Reducing the Dimensionality of Data with Neural Networks», *Science*, вип. 313, вип. 5786, с. 504–507, Лип 2006, doi: 10.1126/science.1127647.

[78] S. Li, «Time Series of Price Anomaly Detection with LSTM», *Medium*, Вер 08, 2020. https://towardsdatascience.com/time-series-of-price-anomaly-detection-with-lstm-11a12ba4f6d9 (дата звернення Бер 26, 2021).

[79] «Introduction to autoencoders.», *Jeremy Jordan*, Бер 19, 2018. https://www.jeremyjordan.me/autoencoders/ (дата звернення Бер 26, 2021).

[80] M. Maas, «A Taxonomy of ML for Systems Problems», *IEEE Micro*, вип. 40, вип. 5, с. 8–16, Вер 2020, doi: 10.1109/MM.2020.3012883.

[81] «scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation». https://scikit-learn.org/stable/ (дата звернення Бер 26, 2021).

[82] «What is Supervised Learning?», Сер 28, 2020. https://www.ibm.com/cloud/learn/supervised-learning (дата звернення Бер 26, 2021).

[83] «Unsupervised Machine Learning: What is, Algorithms, Example». https://www.guru99.com/unsupervised-machine-learning.html (дата звернення Бер 26, 2021).

[84] R. Agrawal, «Fast Algorithms for Mining Association Rules», с. 13.

[85] «Deep learning for time series classification: a review | SpringerLink». https://link.springer.com/article/10.1007%2Fs10618-019-00619-1 (дата звернення Бер 26, 2021).

[86] A. Amidon, «A Brief Survey of Time Series Classification Algorithms», *Medium*, Вер 27, 2020. https://towardsdatascience.com/a-brief-introduction-to-time-series-classification-algorithms-7b4284d31b97 (дата звернення Бер 26, 2021).

[87] A. Abanda, U. Mori, і J. A. Lozano, «A review on distance based time series classification», *Data Min. Knowl. Discov.*, вип. 33, вип. 2, с. 378–412, Бер 2019, doi: 10.1007/s10618-018-0596-4.

[88] E. Fix, *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*. USAF School of Aviation Medicine, 1985.

[89] «Dynamic Time Warping | SpringerLink». https://link.springer.com/chapter/10.1007/978-3-540-74048-3_4 (дата звернення Бер 26, 2021).

[90] «Dynamic Time Warping — tslearn 0.5.0.5 documentation». https://tslearn.readthedocs.io/en/stable/user_guide/dtw.html (дата звернення Бер 26, 2021).

[91] «Time series shapelets: a novel technique that allows accurate, interpretable and fast classification | SpringerLink». https://link.springer.com/article/10.1007/s10618-010-0179-5 (дата звернення Бер 26, 2021).

[92] H. Deng, G. Runger, E. Tuv, і M. Vladimir, «A Time Series Forest for Classification and Feature Extraction», *ArXiv13022277 Cs*, Лют 2013, Дата звернення: Бер 26, 2021. [Online]. Доступний у: http://arxiv.org/abs/1302.2277.

[93] M. Middlehurst, J. Large, і A. Bagnall, «The Canonical Interval Forest (CIF) Classifier for Time Series Classification», *ArXiv200809172 Cs Eess*, Сер 2020, Дата звернення: Бер 26, 2021. [Online]. Доступний у: http://arxiv.org/abs/2008.09172.

[94] A. Bagnall, A. Bostrom, J. Large, і J. Lines, «The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version», Лют 2016.

[95] «Efficient Visual Search of Videos Cast as Text Retrieval | IEEE Journals & Magazine | IEEE Xplore». https://ieeexplore.ieee.org/document/4509438 (дата звернення Бер 26, 2021).

[96] P. Schäfer і M. Högqvist, «SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets», в *Proceedings of the 15th International Conference on Extending Database Technology - EDBT '12*, Berlin, Germany, 2012, с. 516, doi: 10.1145/2247596.2247656.

[97] G. Ke *et al.*, «LightGBM: A Highly Efficient Gradient Boosting Decision Tree», *Adv. Neural Inf. Process. Syst.*, вип. 30, 2017, Дата звернення: Бер 26, 2021. [Online]. Доступний у: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

[98] «The Contract Random Interval Spectral Ensemble (c-RISE): The Effect of Contracting a Classifier on Accuracy | SpringerLink». https://link.springer.com/chapter/10.1007/978-3-030-29859-3_33 (дата звернення Бер 26, 2021).

[99] M. Arul і A. Kareem, «Data Anomaly Detection for Structural Health Monitoring of Bridges using Shapelet Transform», *ArXiv200900470 Cs Eess*, Сер 2020,

Дата звернення: Бер 26, 2021. [Online]. Доступний у: http://arxiv.org/abs/2009.00470.

[100] I. Goodfellow, P. McDaniel, і N. Papernot, «Making machine learning robust against adversarial inputs», *Commun. ACM*, вип. 61, вип. 7, с. 56–66, Чер 2018, doi: 10.1145/3134599.

[101] S. Mitra і S. K. Pal, «Fuzzy multi-layer perceptron, inferencing and rule generation», *IEEE Trans. Neural Netw.*, вип. 6, вип. 1, с. 51–63, Січ 1995, doi: 10.1109/72.363450.

[102] P. Mathur, «A Simple Multilayer Perceptron with TensorFlow», *Medium*, Лют 06, 2018. https://medium.com/pankajmathur/a-simple-multilayer-perceptron-with-tensorflow-3effe7bf3466 (дата звернення Бер 26, 2021).

[103] Uniqtech, «Understand the Softmax Function in Minutes», *Medium*, Жов 06, 2020. https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d (дата звернення Бер 26, 2021).

[104] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, і R. Fergus, «Regularization of Neural Networks using DropConnect», в *International Conference on Machine Learning*, Трав 2013, с. 1058–1066, Дата звернення: Квіт 08, 2021. [Online]. Доступний у: http://proceedings.mlr.press/v28/wan13.html.

[105] N. Mboga, C. Persello, J. R. Bergado, і A. Stein, «Detection of Informal Settlements from VHR Images Using Convolutional Neural Networks», *Remote Sens.*, вип. 9, с. 1106, Жов 2017, doi: 10.3390/rs9111106.

[106] Y. Lecun, L. Bottou, Y. Bengio, і P. Haffner, «Gradient-based learning applied to document recognition», *Proc. IEEE*, вип. 86, вип. 11, с. 2278–2324, Лис 1998, doi: 10.1109/5.726791.

[107] P. Radhakrishnan, «What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?», *Medium*, Жов 18, 2017. https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a (дата звернення Квіт 08, 2021).

[108] A. J. Lockett і R. Miikkulainen, «Temporal Convolution Machines for Sequence Learning», с. 8.

[109] «Temporal Convolutional Networks and Forecasting | by Francesco Lässig | Unit8 - Big Data & AI | Medium». https://medium.com/unit8-machine-learning-publication/temporal-convolutional-networks-and-forecasting-5ce1b6e97ce4 (дата звернення Квіт 08, 2021).

[110] S. Bai, V. Koltun, і J. Z. Kolter, «Multiscale Deep Equilibrium Models», *ArXiv200608656 Cs Stat*, Лис 2020, Дата звернення: Квіт 08, 2021. [Online]. Доступний у: http://arxiv.org/abs/2006.08656.

[111] «Papers with Code - Dilated Convolution Explained». https://paperswithcode.com/method/dilated-convolution (дата звернення Квіт 08, 2021).

[112] N. Arbel, «How LSTM networks solve the problem of vanishing gradients», *Medium*, Трав 16, 2020. https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577 (дата звернення Квіт 08, 2021).

[113] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, і H. Arshad, «State-of-the-art in artificial neural network applications: A survey», *Heliyon*, вип. 4, вип. 11, Лис 2018, doi: 10.1016/j.heliyon.2018.e00938.

[114] T. Zebin, N. Peek, A. Casson, і M. Sperrin, «Human activity recognition from inertial sensor time-series using batch normalized deep LSTM recurrent networks», Лип 2018, вип. 2018, doi: 10.1109/EMBC.2018.8513115.

[115] «Echo State Property of Deep Reservoir Computing Networks | SpringerLink». https://link.springer.com/article/10.1007/s12559-017-9461-9 (дата звернення Бер 26, 2021).

[116] H. Jaeger і H. Haas, «Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication», *Science*, вип. 304, вип. 5667, с. 78–80, Квіт 2004, doi: 10.1126/science.1091277.

[117] H. Jaeger, «Echo state network», *Scholarpedia*, вип. 2, вип. 9, с. 2330, Вер 2007, doi: 10.4249/scholarpedia.2330.

[118] M. Phi, «Illustrated Guide to LSTM’s and GRU’s: A step by step explanation», *Medium*, Чер 28, 2020. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (дата звернення Квіт 08, 2021).

[119] D. Burr, «Speech Recognition Experiments with Perceptrons», в *Neural Information Processing Systems*, 1988, Дата звернення: Квіт 08, 2021. [Online]. Доступний у: https://proceedings.neurips.cc/paper/1987/file/2a38a4a9316c49e5a833517c45d31070-Paper.pdf.

[120] «Understanding LSTM Networks -- colah’s blog». https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення Квіт 08, 2021).

[121] M. Grogan, «CNN-LSTM: Predicting Daily Hotel Cancellations», *Medium*, Січ 07, 2021. https://towardsdatascience.com/cnn-lstm-predicting-daily-hotel-cancellations-e1c75697f124 (дата звернення Квіт 08, 2021).

[122] Md. Z. Islam, Md. M. Islam, і A. Asraf, «A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images», *Inform. Med. Unlocked*, вип. 20, с. 100412, Січ 2020, doi: 10.1016/j.imu.2020.100412.

[123] A. Vaswani *et al.*, «Attention Is All You Need», *ArXiv170603762 Cs*, Груд 2017, Дата звернення: Квіт 08, 2021. [Online]. Доступний у: http://arxiv.org/abs/1706.03762.

[124] «Attention? Attention!» https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html (дата звернення Квіт 08, 2021).

[125] «Attention Mechanism In Deep Learning | Attention Model Keras», *Analytics Vidhya*, Лис 20, 2019. https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/ (дата звернення Квіт 08, 2021).

[126] D. Bahdanau, K. Cho, і Y. Bengio, «Neural Machine Translation by Jointly Learning to Align and Translate», *ArXiv14090473 Cs Stat*, Трав 2016, Дата

звернення: Квіт 08, 2021. [Online]. Доступний у: http://arxiv.org/abs/1409.0473.

[127] J. Cheng, L. Dong, і M. Lapata, «Long Short-Term Memory-Networks for Machine Reading», *ArXiv160106733 Cs*, Вер 2016, Дата звернення: Квіт 08, 2021. [Online]. Доступний у: http://arxiv.org/abs/1601.06733.

[128] S. J. Pan і Q. Yang, «A Survey on Transfer Learning», *IEEE Trans. Knowl. Data Eng.*, вип. 22, вип. 10, с. 1345–1359, Жов 2010, doi: 10.1109/TKDE.2009.191.

[129] J. Serrà, S. Pascual, і A. Karatzoglou, «Towards a universal neural network encoder for time series», *ArXiv180503908 Cs Stat*, Трав 2018, Дата звернення: Бер 26, 2021. [Online]. Доступний у: http://arxiv.org/abs/1805.03908.

[130] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, і P.-A. Muller, «Transfer learning for time series classification», *2018 IEEE Int. Conf. Big Data Big Data*, с. 1367–1376, Груд 2018, doi: 10.1109/BigData.2018.8621990.

[131] X. Glorot і Y. Bengio, «Understanding the difficulty of training deep feedforward neural networks», в *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Бер 2010, с. 249–256, Дата звернення: Бер 26, 2021. [Online]. Доступний у: http://proceedings.mlr.press/v9/glorot10a.html.

[132] J. Yosinski, J. Clune, Y. Bengio, і H. Lipson, «How transferable are features in deep neural networks?», *ArXiv14111792 Cs*, Лис 2014, Дата звернення: Бер 26, 2021. [Online]. Доступний у: http://arxiv.org/abs/1411.1792.

[133] W. Pan, N. Liu, E. Xiang, і Q. Yang, «Transfer Learning to Predict Missing Ratings via Heterogeneous User Feedbacks», Січ 2011, с. 2318–2323, doi: 10.5591/978-1-57735-516-8/IJCAI11-386.

[134] T. Penzel, G. B. Moody, R. G. Mark, A. L. Goldberger, і J. H. Peter, «Apnea-ECG Database». physionet.org, 2000, doi: 10.13026/C23W2R.

[135] «Welcome to the UCR Time Series Classification/Clustering Page». http://www.cs.ucr.edu/~eamonn/time_series_data/ (дата звернення Квіт 01, 2021).

[136] G. A. Susto, A. Cenedese, і M. Terzi, «Time-Series Classification Methods: Review and Applications to Power Systems Data», в *Big Data Application in Power Systems*, 2018, с. 179–220.

[137] «WDS Linked router network - DD-WRT Wiki». https://wiki.dd-wrt.com/wiki/index.php?title=WDS_Linked_router_network (дата звернення Квіт 08, 2021).

[138] «Learn about WiFi standards and the latest WiFi 6 (802.11 ax)». https://www.netspotapp.com/explaining-wifi-standards.html (дата звернення Квіт 08, 2021).

[139] «Bluetooth Low Energy | Bluetooth Technology Website». https://web.archive.org/web/20170310111443/https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy (дата звернення Квіт 08, 2021).

[140] *RFID Forecasts, Players and Opportunities 2019-2029*. 2019.

[141] «Chapter 1: Service Oriented Architecture (SOA)». https://web.archive.org/web/20170707052149/https://msdn.microsoft.com/en-us/library/bb833022.aspx (дата звернення Бер 26, 2021).

[142] «Microservice Architecture pattern». https://microservices.io/patterns/microservices.html (дата звернення Бер 26, 2021).

[143] L. Flower, «Stream and Event Processing using Apache Spark», *AlphaZetta*. https://alphazetta.ai/academy/training/stream-and-event-processing-using-apache-spark/ (дата звернення Квіт 08, 2021).

[144] «What Is A Data Pipeline? Considerations & Examples», *Hazelcast*. https://hazelcast.com/glossary/data-pipeline/ (дата звернення Квіт 08, 2021).

[145] A. Henao, «Building a Data Warehouse Pipeline: Basic Concepts & Roadmap», *Medium*, Жов 24, 2020. https://towardsdatascience.com/building-a-data-warehouse-pipeline-basic-concepts-roadmap-d14032890ab6 (дата звернення Квіт 08, 2021).

[146] N. Ramapuram, «The Best Data Processing Architectures: LAMBDA VS KAPPA- Tech Primer for Engineering Manager», *Medium*, Вер 23, 2020. https://medium.com/@ramapuram.nagesh/the-best-data-processing-architectures-

lambda-vs-kappa-tech-primer-for-engineering-manager-dd2e67bda5ad (дата звернення Квіт 08, 2021).

[147] «A real-time architecture using Hadoop and Storm » λ lambda-architecture.net». http://lambda-architecture.net/architecture/2013-12-11-a-real-time-architecture-using-hadoop-and-storm-devoxx (дата звернення Квіт 08, 2021).

[148] O. A. Malusare, «Real-Time Data Processing With Lambda Architecture», Master of Science, San Jose State University, San Jose, CA, USA, 2019.

[149] S. Hussain, «4 big data architectures, Data Streaming, Lambda architecture, Kappa architecture and Unifield…», *Medium*, Лют 26, 2021. https://medium.com/dataprophet/4-big-data-architectures-data-streaming-lambda-architecture-kappa-architecture-and-unifield-d9bcbf711eb9 (дата звернення Квіт 08, 2021).

[150] A. Chaugule, «Designing Production-Ready Kappa Architecture for Timely Data Stream Processing», *Uber Engineering Blog*, Січ 23, 2020. https://eng.uber.com/kappa-architecture-data-stream-processing/ (дата звернення Квіт 08, 2021).

[151] «What is a Data Lake?», *SearchAWS*. https://searchaws.techtarget.com/definition/data-lake (дата звернення Квіт 08, 2021).

[152] K. Singh, R. Behera, і J. Mantri, «Big Data Ecosystem: Review on Architectural Evolution: Proceedings of IEMIS 2018, Volume 2», 2019, с. 335–345.

[153] «What Is Lambda Architecture», *Databricks*. https://databricks.com/glossary/lambda-architecture (дата звернення Квіт 08, 2021).

[154] S. Alasmari і M. Anwar, «Security & Privacy Challenges in IoT-Based Health Cloud», Груд 2016, с. 198–201, doi: 10.1109/CSCI.2016.0044.

[155] E. H.-L. U. 27 Dec’18 2018-12-27T08:37:12+00:00, «What is Edge Computing: The Network Edge Explained», *Cloudwards*, Груд 31, 2018. https://www.cloudwards.net/what-is-edge-computing/ (дата звернення Бер 26, 2021).

[156] «ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems». https://www.espressif.com/en/products/socs/esp32 (дата звернення Квіт 08, 2021).

[157] «TensorFlow Lite for Microcontrollers», *TensorFlow*. https://www.tensorflow.org/lite/microcontrollers (дата звернення Квіт 08, 2021).

[158] «TensorFlow». https://www.tensorflow.org/ (дата звернення Квіт 08, 2021).

[159] Q. Khan, «Introduction to Tensorflow Lite», *Medium*, Лис 03, 2020. https://medium.com/analytics-vidhya/introduction-to-tensorflow-lite-66c19d757b8c (дата звернення Квіт 08, 2021).

[160] «Keras: the Python deep learning API». https://keras.io/ (дата звернення Квіт 08, 2021).

[161] R. Ciobotariu, C. Rotariu, F. Adochiei, і H. Costin, «Wireless breathing system for long term telemonitoring of respiratory activity», в *2011 7TH INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE)*, Трав 2011, с. 1–4.

[162] M. Zeng *et al.*, «Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors», представлена на Proceedings of the 2014 6th International Conference on Mobile Computing, Applications and Services, MobiCASE 2014, Лис 2014, doi: 10.4108/icst.mobicase.2014.257786.

[163] F. J. Ordóñez і D. Roggen, «Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition», *Sensors*, вип. 16, вип. 1, Art. вип. 1, Січ 2016, doi: 10.3390/s16010115.

ДОДАТОК А

ИТ-ТРАНЗИТ
інжинірингові та інформаційні послуги у нафтогазовій галузі

Україна, 03150, м. Київ, вул. К. Малевича, 86-Е, оф.8,
тел/факс: +380 (44) 200-81-58, тел.: +380 (44) 390-74-98
86-E, Kazimir Malevich Str., of. 8, 03150-UA, Kyiv, Ukraine,
tel/fax: +380 (44) 200-81-58, tel: +380 (44) 390-74-98
mailbox@it-transit.com.ua, www.it-transit.com

ISO 9001 CERTIFIED

ЄДРПОУ 34492678, р/р UA333206490000026003052666511 в АТ КБ "ПРИВАТБАНК", МФО 320649
EDRPOU 34492678, SA UA333206490000026003052666511 JSC CB "PRYVATBANK", MFO 320649

Вих. № 06-1/2021 від 19 лютого 2021р.

Довідка

про впровадження результатів наукових досліджень в практику діяльності організації

У дисертаційній роботі асистента кафедри системного проектування ННК ІПСА КПІ ім. Ігоря Сікорського Кислого Романа Володимировича на тему "Розпізнавання людських активних дій за допомогою методів штучного інтелекту" запропоновано методи збору та очистки даних з натільних сенсорів, їх попередньої обробки, а також архітектуру моделей машинного навчання і використання зібраних даних для розпізнавання людської діяльності.

Практичну цінність мають методи попередньої обробки даних, отриманих з натільних сенсорів, а також рекомендації що до їх зберігання та використання в моделях машинного навчання на мобільних приладах.

Проведений аналіз та розроблені на його базі пропозиції щодо використання даних отриманих з сенсорів, їх попередня обробка та фільтрація від шумів будуть використані в подальшій практичній діяльності підприємства.

Директор
ТОВ «ИТ-ТРАНЗИТ»
М.П.

А.Г.Михайленко

«19» лютого 2021р.

ДОДАТОК Б

Список публікацій

1. Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018). Detection of human respiration patterns using deep convolution neural networks. Eastern-European Journal of Enterprise Technologies, 4(9 (94)), 6–13. https://doi.org/10.15587/1729-4061.2018.139997 – здобувачем було спроектовано та розроблено BSN систему обробки даних з натільних сенсорів, створено модель машинного навчання для класифікації видів дихання.

2. I. Pysmennyi, A. Petrenko, and R. Kyslyi, (2020) Graph-based fog computing network model, Applied Computer Science, vol. 16, no. 4, pp. 5-20, 2020, doi: 10.23743/acs-2020-25. – здобувачем було розроблено алгоритм пошуку маршруту між прикінцевим вузлом та хмарою в динамічній крайовій мережі.

3. Roman V. Kyslyi, Anatolii I. Petrenko,/ (2020) Розпізнавання людської діяльності за допомогою портативних натільних датчиків, Системні дослідження та інформаційні технології, no. 2,pp. 41-54 2020, doi:https://doi.org/10.20535/SRIT.2308-8893.2020.2.03 . – здобувачем було досліджено використання моделей машинного навчання в розпізнаванні людської діяльності.

4. Petrenko, A., Kyslyi, R., & Pysmennyi, I. (2018). Designing security of personal data in distributed health care platform. Technology Audit and Production Reserves, 2(42). https://doi.org/10.15587/2312-8372.2018.141299 – здобувачем досліджувалися безпека та анонімізаціїя даних пацієнта, передача їх в хмару.

5. Pysmennyi, I., Kyslyi, R., & Petrenko, A. (2019). Edge computing in multi-scope service-oriented mobile healthcare systems. System Research and Information Technologies, 0(1), 118–127. https://doi.org/10.20535/SRIT.2308-8893.2019.1.09 – здобувачем досліджувалася інтеграція прикінцевих обчислень в хмарні системи, досліджувалось використання даних на прикінцевих вузлах мережі для машинного навчання.

ДОДАТОК В

Розробка згорткової нейронної мережі для класифікації активності

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import tensorflow as tf
%matplotlib inline
plt.style.use('ggplot')
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
def read_data(file_path):
    column_names = ['activity','timestamp', 'x', 'y', 'z']
    data = pd.read_csv(file_path,header = None, names = column_names)
    return data

def feature_normalize(dataset):
    mu = np.mean(dataset,axis = 0)
    sigma = np.std(dataset,axis = 0)
    return (dataset - mu)/sigma

def plot_axis(ax, x, y, title):
    ax.plot(x, y)
    ax.set_title(title)
    ax.xaxis.set_visible(False)
    ax.set_ylim([min(y) - np.std(y), max(y) + np.std(y)])
    ax.set_xlim([min(x), max(x)])
    ax.grid(True)

def plot_activity(activity,data):
    fig, (ax0, ax1, ax2) = plt.subplots(nrows = 3, figsize = (15, 10), sharex = True)
    plot_axis(ax0, data['timestamp'], data['x'], 'x')
    plot_axis(ax1, data['timestamp'], data['y'], 'y')
    plot_axis(ax2, data['timestamp'], data['z'], 'z')
```

```
    plt.subplots_adjust(hspace=0.2)
    fig.suptitle(activity)
    plt.subplots_adjust(top=0.90)
    plt.show()
   dataset = read_data('/home/darth/Documents/activity.csv')
dataset['x'] = feature_normalize(dataset['x'])
dataset['y'] = feature_normalize(dataset['y'])
dataset['z'] = feature_normalize(dataset['z'])
for activity in np.unique(dataset["activity"]):
    subset = dataset[dataset["activity"] == activity][:180]
    plot_activity(activity,subset)
def windows(data, size):
    start = 0
    while start < data.count():
        yield int(start), int(start + size)
        start += (size / 2)

def segment_signal(data,window_size = 90):
    segments = np.empty((0,window_size,3))
    labels = np.empty((0))
    for (start, end) in windows(data["timestamp"], window_size):
        x = data["x"][start:end]
        y = data["y"][start:end]
        z = data["z"][start:end]
        if(len(dataset["timestamp"][start:end]) == window_size):
            segments = np.vstack([segments,np.dstack([x,y,z])])
            labels = np.append(labels,stats.mode(data["activity"][start:end])[0][0])
    return segments, labels
def read_data1(file_path):
    column_names = ['activity','timestamp', 'x', 'y', 'z']
    data = pd.read_csv(file_path,header = None,delimiter='|', names = column_names)
    return data

df1 = read_data1('/home/darth/Documents/qq.csv')
df1['x'] = feature_normalize(dataset['x'])
```

df1['y'] = feature_normalize(dataset['y'])

df1['z'] = feature_normalize(dataset['z'])

activity	timestamp	x	y	z	SMA_10	SMA_20	
0	d	491059623260000	-0.431963	0.650520	0.023651	-0.431963	-0.431963
1	d	491060622710000	0.373243	0.434736	0.132313	-0.029360	-0.029360
2	d	491061121670000	0.357869	0.376640	-0.117939	0.099716	0.099716
3	d	491062223050000	-0.420433	1.536478	0.632817	-0.030321	-0.030321
4	d	491063322900000	-0.501145	0.563376	1.643703	-0.124486	-0.124486
...	...	...	...	...	...	...	...
592	d	491689124290000	-0.149468	-0.797721	-0.246358	-0.655268	-0.790078
593	d	491837895350000	-0.491537	-0.878640	1.854442	-0.699660	-0.729351
594	d	491838747100000	-0.474241	-0.820544	1.910419	-0.566100	-0.746070
595	d	491839323570000	-0.485771	-0.756224	1.847856	-0.584548	-0.661514
596	d	491840423120000	-0.485771	-0.486494	1.567969	-0.478084	-0.688802

597 rows × 7 columns

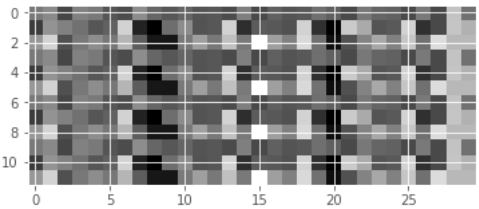
m = [dataset['x'][100:130],dataset['y'][100:130],dataset['z'][100:130],

```
   dataset['x'][100:130],dataset['y'][100:130],dataset['z'][100:130],
   dataset['x'][100:130],dataset['y'][100:130],dataset['z'][100:130],
   dataset['x'][100:130],dataset['y'][100:130],dataset['z'][100:130]]
import numpy as np
print(np.matrix(m))
[[-0.03608615  0.25986056 -1.05844752  0.0926699  -0.04185132 -0.7
7210946
  -0.95467397  0.33672986 -0.43196292 -0.73367485 -0.44733677 -0.7
8748332
  -0.73944002 -0.9662044   0.06000044 -1.00463905 -0.55495377 -0.7
8748332
  -0.70677061 -0.25324185 -0.54918858 -1.00463905 -0.08028598 -0.4
4733677
  -0.50691047 -0.98734342 -0.05338172 -0.98734342  1.40136935 -0.1
9366816]
 [-1.2396627   0.46378389  0.090312   -0.18979188 -0.74999964 -0.5
4044047
   1.6941659  -1.78327171 -2.44307192 -0.28315985  0.31232022 -0.8
1431978
  -0.73340089  1.69001616 -1.54881439 -0.64003296 -0.70850277 -0.8
3091853
   1.70039057 -1.50939235 -2.45967071  1.54270223  0.91402486 -0.7
85272
  -0.67323041  1.70039057 -1.49071878 -0.98860665  1.44310984  1.0
4059028]
 [ 0.54061875  1.74577974 -0.86540248 -0.07842575  0.3265874  -0.1
0806086
   1.12344249 -0.61515047 -1.98824375 -2.0244645  -0.04220506  0.6
8879423
  -0.07184017  1.34735213 -0.05208343  2.59531938  0.68879423  0.0
8621374
   1.83797775 -0.1903806  -2.17263998 -0.84893852  0.16194791  0.8
6660492
   0.0960921   1.43955033 -0.23647965  1.90383379  1.1892983   1.1
892983 ]
 [-0.03608615  0.25986056 -1.05844752  0.0926699  -0.04185132 -0.7
7210946
  -0.95467397  0.33672986 -0.43196292 -0.73367485 -0.44733677 -0.7
8748332
  -0.73944002 -0.9662044   0.06000044 -1.00463905 -0.55495377 -0.7
8748332
  -0.70677061 -0.25324185 -0.54918858 -1.00463905 -0.08028598 -0.4
4733677
  -0.50691047 -0.98734342 -0.05338172 -0.98734342  1.40136935 -0.1
9366816]
 [-1.2396627   0.46378389  0.090312   -0.18979188 -0.74999964 -0.5
4044047
   1.6941659  -1.78327171 -2.44307192 -0.28315985  0.31232022 -0.8
1431978
```

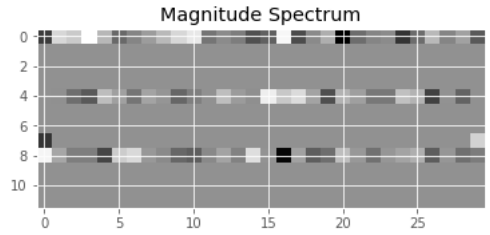
```
  -0.73340089  1.69001616 -1.54881439 -0.64003296 -0.70850277 -0.8
3091853
   1.70039057 -1.50939235 -2.45967071  1.54270223  0.91402486 -0.7
85272
  -0.67323041  1.70039057 -1.49071878 -0.98860665  1.44310984  1.0
4059028]
 [ 0.54061875  1.74577974 -0.86540248 -0.07842575  0.3265874  -0.1
0806086
   1.12344249 -0.61515047 -1.98824375 -2.0244645  -0.04220506  0.6
8879423
  -0.07184017  1.34735213 -0.05208343  2.59531938  0.68879423  0.0
8621374
   1.83797775 -0.1903806  -2.17263998 -0.84893852  0.16194791  0.8
6660492
   0.0960921   1.43955033 -0.23647965  1.90383379  1.1892983   1.1
892983 ]
 [-0.03608615  0.25986056 -1.05844752  0.0926699  -0.04185132 -0.7
7210946
  -0.95467397  0.33672986 -0.43196292 -0.73367485 -0.44733677 -0.7
8748332
  -0.73944002 -0.9662044   0.06000044 -1.00463905 -0.55495377 -0.7
8748332
  -0.70677061 -0.25324185 -0.54918858 -1.00463905 -0.08028598 -0.4
4733677
  -0.50691047 -0.98734342 -0.05338172 -0.98734342  1.40136935 -0.1
9366816]
 [-1.2396627   0.46378389  0.090312   -0.18979188 -0.74999964 -0.5
4044047
   1.6941659  -1.78327171 -2.44307192 -0.28315985  0.31232022 -0.8
1431978
  -0.73340089  1.69001616 -1.54881439 -0.64003296 -0.70850277 -0.8
3091853
   1.70039057 -1.50939235 -2.45967071  1.54270223  0.91402486 -0.7
85272
  -0.67323041  1.70039057 -1.49071878 -0.98860665  1.44310984  1.0
4059028]
 [ 0.54061875  1.74577974 -0.86540248 -0.07842575  0.3265874  -0.1
0806086
   1.12344249 -0.61515047 -1.98824375 -2.0244645  -0.04220506  0.6
8879423
  -0.07184017  1.34735213 -0.05208343  2.59531938  0.68879423  0.0
8621374
   1.83797775 -0.1903806  -2.17263998 -0.84893852  0.16194791  0.8
6660492
   0.0960921   1.43955033 -0.23647965  1.90383379  1.1892983   1.1
892983 ]
 [-0.03608615  0.25986056 -1.05844752  0.0926699  -0.04185132 -0.7
7210946
  -0.95467397  0.33672986 -0.43196292 -0.73367485 -0.44733677 -0.7
8748332
  -0.73944002 -0.9662044   0.06000044 -1.00463905 -0.55495377 -0.7
8748332
```

```
  -0.70677061 -0.25324185 -0.54918858 -1.00463905 -0.08028598 -0.4
4733677
  -0.50691047 -0.98734342 -0.05338172 -0.98734342  1.40136935 -0.1
9366816]
 [-1.2396627   0.46378389  0.090312   -0.18979188 -0.74999964 -0.5
4044047
   1.6941659  -1.78327171 -2.44307192 -0.28315985  0.31232022 -0.8
1431978
  -0.73340089  1.69001616 -1.54881439 -0.64003296 -0.70850277 -0.8
3091853
   1.70039057 -1.50939235 -2.45967071  1.54270223  0.91402486 -0.7
85272
  -0.67323041  1.70039057 -1.49071878 -0.98860665  1.44310984  1.0
4059028]
 [ 0.54061875  1.74577974 -0.86540248 -0.07842575  0.3265874  -0.1
0806086
   1.12344249 -0.61515047 -1.98824375 -2.0244645  -0.04220506  0.6
8879423
  -0.07184017  1.34735213 -0.05208343  2.59531938  0.68879423  0.0
8621374
   1.83797775 -0.1903806  -2.17263998 -0.84893852  0.16194791  0.8
6660492
   0.0960921   1.43955033 -0.23647965  1.90383379  1.1892983   1.1
892983 ]]
```

```
from matplotlib import pyplot as plt
plt.imshow(np.matrix(m))
plt.gray()
plt.savefig('h.tiff',dpi=300)
```




```
import cv2
dft = cv2.dft(np.matrix(m), flags=cv2.DFT_SCALE | cv2.DFT_REAL_OUTPUT)#,flags =
cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
plt.imshow(dft)
plt.title('Magnitude Spectrum')#, plt.xticks([]), plt.yticks([])
plt.gray()
plt.savefig('h.tiff',dpi=300)
```




```
segments, labels = segment_signal(dataset)
labels = np.asarray(pd.get_dummies(labels), dtype = np.int8)
reshaped_segments = segments.reshape(len(segments), 1,90, 3)
train_test_split = np.random.rand(len(reshaped_segments)) < 0.70
train_x = reshaped_segments[train_test_split]
train_y = labels[train_test_split]
test_x = reshaped_segments[~train_test_split]
test_y = labels[~train_test_split]
input_height = 1
input_width = 90
num_labels = 4
num_channels = 3

batch_size = 10
kernel_size = 60
depth = 60
num_hidden = 1000

learning_rate = 0.0001
training_epochs = 100
total_batches = train_x.shape[0] // batch_size
total_batchs = train_x.shape[0] // batch_size

def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev = 0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.0, shape = shape)
    return tf.Variable(initial)
```

```
def depthwise_conv2d(x, W):
    return tf.nn.depthwise_conv2d(x,W, [1, 1, 1, 1], padding='VALID')

def apply_depthwise_conv(x,kernel_size,num_channels,depth):
    weights = weight_variable([1, kernel_size, num_channels, depth])
    biases = bias_variable([depth * num_channels])
    return tf.nn.relu(tf.add(depthwise_conv2d(x, weights),biases))

def apply_max_pool(x,kernel_size,stride_size):
    return tf.nn.max_pool(x, ksize=[1, 1, kernel_size, 1],
                          strides=[1, 1, stride_size, 1], padding='VALID')
X = tf.placeholder(tf.float32, shape=[None,input_height,input_width,num_channels])
Y = tf.placeholder(tf.float32, shape=[None,num_labels])

c = apply_depthwise_conv(X,kernel_size,num_channels,depth)
p = apply_max_pool(c,20,2)
c = apply_depthwise_conv(p,6,depth*num_channels,depth//10)

shape = c.get_shape().as_list()
c_flat = tf.reshape(c, [-1, shape[1] * shape[2] * shape[3]])

f_weights_l1 = weight_variable([shape[1] * shape[2] * depth * num_channels * (depth//10),
num_hidden])
f_biases_l1 = bias_variable([num_hidden])
f = tf.nn.tanh(tf.add(tf.matmul(c_flat, f_weights_l1),f_biases_l1))

out_weights = weight_variable([num_hidden, num_labels])
out_biases = bias_variable([num_labels])
y_ = tf.nn.softmax(tf.matmul(f, out_weights) + out_biases)
loss = -tf.reduce_sum(Y * tf.log(y_))
optimizer = tf.train.GradientDescentOptimizer(learning_rate = learning_rate).minimize(loss)

correct_prediction = tf.equal(tf.argmax(y_,1), tf.argmax(Y,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
cost_history = np.empty(shape=[1],dtype=float)

with tf.Session() as session:
    tf.global_variables_initializer().run()
    tf.local_variables_initializer()
    for epoch in range(training_epochs):
        for b in range(total_batches):
            offset = (b * batch_size) % (train_y.shape[0] - batch_size)
            batch_x = train_x[offset:(offset + batch_size), :, :, :]
            batch_y = train_y[offset:(offset + batch_size), :]
            _, c = session.run([optimizer, loss],feed_dict={X: batch_x, Y : batch_y})
            cost_history = np.append(cost_history,c)
        print("Epoch: ",epoch," Training Loss: ",np.mean(cost_history)," Training Accuracy: ",
              session.run(accuracy, feed_dict={X: train_x, Y: train_y}))
        output = session.run(y_, {X: train_x})
        auc, update_op = tf.metrics.auc(train_y,output)
        session.run(tf.local_variables_initializer())
        print("tf auc: {}".format(session.run([auc, update_op])))

        y_p = tf.argmax(output, 1)
        val_accuracy, y_pred = session.run([accuracy, y_p], feed_dict={X:train_x, Y:train_y})

        print("validation accuracy:", val_accuracy)
        y_true = np.argmax(train_y,1)
        print("Precision", precision_score(y_true, y_pred, average='weighted'))
        print("Recall", recall_score(y_true, y_pred, average='weighted'))
        print("f1_score", f1_score(y_true, y_pred, average='weighted'))
        print('_____________________________________________________________')
        #print("confusion_matrix")
```

```
    #print(sk.metrics.confusion_matrix(y_true, y_pred, average='micro'),fpr, tpr, tresholds =
sk.metrics.roc_curve(y_true, y_pred, average='micro')
    print("Testing Accuracy:", session.run(accuracy, feed_dict={X: test_x, Y: test_y}))

  output = session.run(y_, {X: test_x})
  cm = tf.confusion_matrix(labels=tf.argmax(test_y, axis = 1), predictions=tf.argmax(output,
axis = 1), num_classes=num_labels)
  print("confusion_matrix")
  print(cm.eval())
    Epoch:  0  Training Loss:  8.560974779583159  Training Accurac
y:  0.47217807
tf auc: [1887000000.0, 0.86959934]
validation accuracy: 0.47217807
Precision 0.641485475245066
Recall 0.47217806041335453
f1_score 0.3165350453097321
_____________________________________________________________
Testing Accuracy: 0.48387095
Epoch:  5  Training Loss:  5.536392670631408  Training Accuracy:
0.7106518
tf auc: [0.92612374, 0.92612374]
validation accuracy: 0.7106518
Precision 0.6474695301999134
Recall 0.7106518282988871
f1_score 0.6584604440952072
_____________________________________________________________
Testing Accuracy: 0.6953405
Epoch:  10  Training Loss:  3.828273404728282  Training Accuracy:
0.8155803
tf auc: [244666800.0, 0.95309424]
validation accuracy: 0.8155803
Precision 0.7236439499752193
Recall 0.8155802861685215
f1_score 0.7668653426140417
_____________________________________________________________
Testing Accuracy: 0.8136201
Epoch:  15  Training Loss:  3.301611890395482  Training Accuracy:
0.8426073
tf auc: [1887000000.0, 0.9636135]
validation accuracy: 0.8426073
Precision 0.7531619012643161
Recall 0.8426073131955485
f1_score 0.7942566093902463
_____________________________________________________________
Testing Accuracy: 0.827957
```

```
Epoch:  20  Training Loss:  3.885034371227314  Training Accuracy:
0.84737676
tf auc: [0.9712374, 0.9712374]
validation accuracy: 0.84737676
Precision 0.7590602167143627
Recall 0.8473767885532592
f1_score 0.7991362094337763

_________________________________________________________________
Testing Accuracy: 0.83870965
Epoch:  25  Training Loss:  2.5416351873177945  Training Accuracy:
0.8489666
tf auc: [0.0, 0.97737473]
validation accuracy: 0.8489666
Precision 0.760829191671799
Recall 0.848966613672496
f1_score 0.8007094600801851

_________________________________________________________________
Testing Accuracy: 0.83870965
Epoch:  30  Training Loss:  2.2503983707948665  Training Accuracy:
0.85532594
tf auc: [1887000000.0, 0.9817958]
validation accuracy: 0.85532594
Precision 0.7699418711048714
Recall 0.8553259141494436
f1_score 0.8082069378177065

_________________________________________________________________
Testing Accuracy: 0.83870965
Epoch:  35  Training Loss:  2.998350059422449  Training Accuracy:
0.8648649
tf auc: [1.0000001, 0.98498255]
validation accuracy: 0.8648649
Precision 0.8114421532224714
Recall 0.8648648648648649
f1_score 0.8268542749537547

_________________________________________________________________
Testing Accuracy: 0.8422939
Epoch:  40  Training Loss:  1.776921708383372  Training Accuracy:
0.87599367
tf auc: [0.9872826, 0.9872826]
validation accuracy: 0.87599367
Precision 0.8235870187642527
Recall 0.875993640699523
f1_score 0.842230330485364

_________________________________________________________________
Testing Accuracy: 0.8458781
Epoch:  45  Training Loss:  1.580162515122917  Training Accuracy:
0.9507157
tf auc: [0.9890776, 0.9890776]
validation accuracy: 0.89507157
Precision 0.9011272803038461
```

```
Recall 0.8950715421303657
f1_score 0.8689100763140937
_________________________________________________________________
Testing Accuracy: 0.8734624
Epoch:  50  Training Loss:  1.403763567000671  Training Accuracy:
0.9809698
tf auc: [0.99058497, 0.99058497]
validation accuracy: 0.9109698
Precision 0.9187572705305853
Recall 0.9839697933227345
f1_score 0.9833629265347004
```

Модель автоенкодера для виявлення аномалій:

```
TIME_STEPS = 8

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2,
activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2,
activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2,
activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2,
activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="
same"),
    ]
```

```
)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001)
, loss="mse")
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_8 (Conv1D)            (None, 4, 32)             256
_________________________________________________________________
dropout_8 (Dropout)          (None, 4, 32)             0
_________________________________________________________________
conv1d_9 (Conv1D)            (None, 2, 16)             3600
_________________________________________________________________
conv1d_transpose_12 (Conv1DT (None, 4, 16)             1808
_________________________________________________________________
dropout_9 (Dropout)          (None, 4, 16)             0
_________________________________________________________________
conv1d_transpose_13 (Conv1DT (None, 8, 32)             3616
_________________________________________________________________
conv1d_transpose_14 (Conv1DT (None, 8, 1)              225
=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=32,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience
=5, mode="min")
    ],
)
```

```
Epoch 1/50
4/4 [==============================] - 1s 59ms/step - loss: 1.1553 - val_los
s: 0.2057
Epoch 2/50
4/4 [==============================] - 0s 10ms/step - loss: 1.0746 - val_los
s: 0.2092
Epoch 3/50
4/4 [==============================] - 0s 10ms/step - loss: 1.0672 - val_los
s: 0.2162
Epoch 4/50
4/4 [==============================] - 0s 10ms/step - loss: 1.0317 - val_los
s: 0.2214
Epoch 5/50
4/4 [==============================] - 0s 10ms/step - loss: 0.8650 - val_los
s: 0.2225
Epoch 6/50
4/4 [==============================] - 0s 11ms/step - loss: 0.6616 - val_los
s: 0.2155
```

```
def normalize_test(values, mean, std):
    values -= mean
```

```
    values /= std
    return values


df_test_value = (df_daily_jumpsup - training_mean) / training_std
fig, ax = plt.subplots()
df_test_value.plot(legend=False, ax=ax)
plt.show()

# Create sequences from test values.
x_test = create_sequences(df_test_value.values)
print("Test input shape: ", x_test.shape)

# Get test MAE loss.
x_test_pred = model.predict(x_test)
test_mae_loss = np.mean(np.abs(x_test_pred - x_test), axis=1)
test_mae_loss = test_mae_loss.reshape((-1))

plt.hist(test_mae_loss, bins=50)
plt.xlabel("test MAE loss")
plt.ylabel("No of samples")
plt.show()

# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))
```