МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет
імені В. Н. Каразіна
Факультет радіофізики, біомедичної
електроніки та комп'ютерних систем
Кафедра _____

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____
підпис ініціали, прізвище

"____" _____20___ року

Кваліфікаційна робота магістра

на тему: МЕТОД СЕГМЕНТАЦІЇ ЕКГ НА БАЗІ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

Виконав: студент ІІ курсу магістратури, групи РЕ-61
спеціальності 176 Мікро- та наносистемна техніка
освітньо-професійна програма «Фізична та біомедична електроніка»

Андрій ЖУК.

Керівник
доктор фіз.-мат. наук,
професор Ольга Величко

Консультант
к.ф.-м.н., ст.н.с. Бердник С.М.

20___ рік

АНОТАЦІЯ

Пояснювальна записка атестаційної роботи магістра: 59 с., 16 рис., елменти коду, 21 джерело.

Об’єкт дослідження: Р-хвиля, QRS-комплекс та Т-хвиля

Предмет дослідження: методи сегментації електрокардіограми за допомогою штучної нейромережі.

Мета роботи: розробка ефективної моделі для аналізу ЕКГ-сигналів, яка дозволить автоматично виділяти ключові інтервали та сегменти, такі як Р-хвиля, QRS-комплекс та Т-хвиля, що є важливими для діагностики серцево-судинних захворювань.

Проведено медико-технічне обґрунтування роботи. У роботі проведено аналіз сучасних методів обробки ЕКГ та існуючих підходів до сегментації сигналів. Розробка архітектури нейронної мережі, оптимізованої для задач сегментації. Використання сучасних бібліотек та інструментів машинного навчання для тренування моделі на анотованих наборах даних ЕКГ.

Порівняльний аналіз результатів сегментації, отриманих за допомогою запропонованої моделі, з результатами традиційних методів.

Розроблений модуль може бути застосований як допоміжний програмний засіб при проектуванні кардіологічних систем, функцією яких є автоматичний аналіз даних, а також при вивченні методів обробки та аналізу електрофізіологічних сигналів в рамках лабораторного практикуму в учбовому процесі.

ЕКГ, СЕГМЕНТАЦІЯ, ШТУЧНА НЕЙРОННА МЕРЕЖА, МАШИННЕ НАВЧАННЯ, QRS-КОМПЛЕКС, МЕДИЧНА ДІАГНОСТИКА.

ЗМІСТ

АНОТАЦІЯ 2

ВСТУП 4

1 МЕДИЧНО – ТЕХНІЧНЕ ОБГРУНТУВАННЯ 7

1.1 Фізіологічні основи серцевої діяльності 7

1.2 Электрокардіографія. Система ЭКГ відведень 11

1.3 Методики виділення QRS - комплексів 15

2 АНАЛІЗ МЕТОДІВ ОБРОБКИ ТА СЕГМЕНТАЦІЇ ЕКГ- СИГНАЛІВ 21

2.1 Традиційний алгоритм Пан-Томпкінса 21

2.2 Сучасні підходи, що базуються на методах машинного навчання, пошук QRS комплексів за допомогою нейронної мережі. 24

2.3 Порядок виконання практичної роботи 34

2.4 Створення нейронної мережі на практиці 38

2.5 Тестування мережі 41

3 РОЗРОБКА ПРАКТИЧНОЇ МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ З МАШИННИМ НАВЧАННЯМ ЗА ДОПОМОГОЮ PYTHON 43

ВИСНОВОК 55

СПИСОК ДЖЕРЕЛ ..... 57**Ошибка! Закладка не определена.**

ВСТУП

За даними ВООЗ Україна є першою країною у Європі і другою в світі в рейтингу смертності від серцево – судинних захворювань. З серцево – судинними захворюваннями стикається кожен другий українець. Такого високого показника – 68 % - немає в жодній розвиненій країні світу, а в Європі і Америці ці цифри на порядок нижче. В останнє десятиліття спостерігається значне зниження вікового цензу пацієнтів. Дуже багато дітей народжується з вадами серця. Основні причини, за якими інфаркти та інсульти почали «досягати» українців від 30 до 40 років, - неправильне харчування, малорухливий спосіб життя, стреси та шкідливі звички. На перше січня 2019 року населення України становило 42,2 мільйона осіб. За даними перепису 2001 року, в Україні проживали 48,4 мільйона людей. У 2018 році серцево – судинні захворювання забрали життя 392 тисячі українців. Динаміка зміни чисельності населення України ілюструється наведеним нижче графіком ,яка представлена на рисунку 1.1.

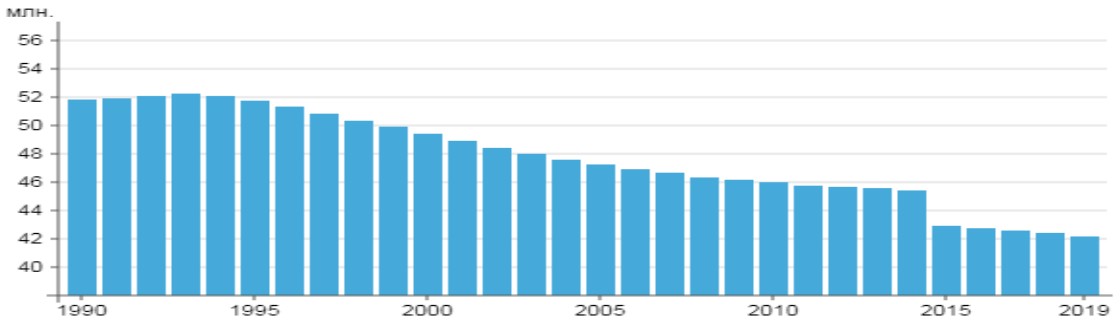


Рисунок 1.1 – Динаміка зміни чисельності населення України

Згідно зі статистикою 2018 року, в структурі поширеності ССЗ відсоток осіб працездатного віку є значним і становить при ішемічній хворобі серця - 27,8 %, порушеннях ритму серця - 31,1 %, інфаркті міокарда - 29,7 %. В цілому, у 37 % працездатного населення України діагностовано ССЗ, кожен

четвертий пацієнт із серцево-судинною патологією має вік від 17 до 69 років, за статево - віковою структурою захворюваності сімдесят відсотків переважає чоловіче населення Згідно зі статистикою Асоціації кардіологів України, близько двадцяти мільйонів дорослих українців страждають на ті чи інші серцево-судинні захворювання. Населення країни мало цікавиться питаннями здоров'я. Споживання в їжу трансгенних жирів, солі, цукру, шкодить організму і стає причиною серцево - судинних захворювань. Недостатньо уваги приділяється екологічним проблем, які теж позначаються на здоров'ї. Спосіб життя і фактори зовнішнього середовища в сумі послаблюють наш організм і роблять його вразливим для хвороб.

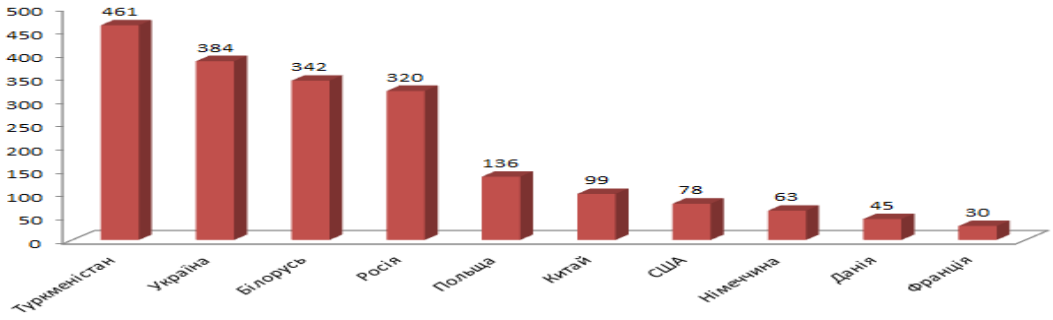


Рисунок 1.2 – Смертність на 100 тисяч осіб від серцево – судинних хвороб у різних країнах

На сьогоднішній день в нашій країні є медичні центри із сучасним устаткуванням, яке може виявити та з максимальною чіткістю зафіксувати найменші порушення в роботі серця та судин для правильного встановлення діагнозу та результативного лікування. Одним з найефективніших інструментів боротьби з серцево - судинними захворюваннями є рання діагностика. Надання висококваліфікованої медичної допомоги, зниження смертності, покращення якості життя, мають надзвичайно велике значення. Сучасні технології, на основі яких створюється діагностичне обладнання, дозволять максимально точно поставити діагноз та призначити ефективне лікування. Сучасна медична діагностика базується на

доказовому підході, який заснований на використанні високоточної апаратури та нових інформаційних технологій для отримання достовірних даних про стан серцево – судинної системи людини .

Аналіз досліджень та публікацій. Отримані числові значення ВСР оцінюються різними дослідниками в залежності від використовуваної наукової концепції. Аналіз заснований на фізіологічній інтерпретації показників, одержуваних за допомогою спеціального графіка - інтервальної гістограми належить Баєвському Р. М. Парін В. В. розглядав коливання тривалості кардіоінтервалів як результат впливу багаторівневої системи управління фізіологічними функціями організму, Анохін П. К. як теорії функціональних систем.

Візуальний аналіз кардіоінтервалограм (ритмограм) був введений Д. І. Жемайтіте. Запропонована нею класифікація ритмограм не втратила своєї актуальності [33]. Дослідники Миронова Т. В. та Миронов В. А. займалися дослідженням на основі комп'ютерного анализа коротких ділянок загальної варіабельності, періодичними складовими та дослідженнями внутрішньої організації динамічного ряду кардіоінтервалів. Із застосуванням візуального, тимчасового і спектрального видів аналіза розробили наступне застосування даного методу дослідження ВРС при аритміях:

-для визначення вегетативного фону порушень ритму серця;

-з метою виявлення додаткових ознак електричної нестабільності міокарда.

У базі даних PubMed на 2019 рік з тегом варіабельність серцевого ритму припадає 23875 публікацій, є також з допомогою смартфона, оскільки таке вимірювання є економічно обгрунтованою, доцільною і порівняльною методологією з автономними системами девайсів та датчиків. В останні роки стосовно оцінки ВСР науковці навіть впроваджують новий термін варіабельність частоти пульсу. Коли нам потрібен метод швидкого аналізу та

інтерпретації отриманих результатів та ми знаходимось далеко від лікарні, використання смартфона може бути дуже ефективним, а можливість самостійного використання є дуже важливою. В Україні метод знаходить все більш широке застосування .

1 МЕДИЧНО – ТЕХНІЧНЕ ОБГРУНТУВАННЯ

1.1 Фізіологічні основи серцевої діяльності

Збудження серця починається у синусовому вузлі, який розташований у правому передсерді в області гирла верхньої порожнистої вени (рисунок 1.3). Синусовий вузол має властивість автоматизму і генерує певну кількість імпульсів за визначений проміжок часу. У дорослої людини в стані спокою в синусовому вузлі утворюється 60–80 імпульсів за хвилину.

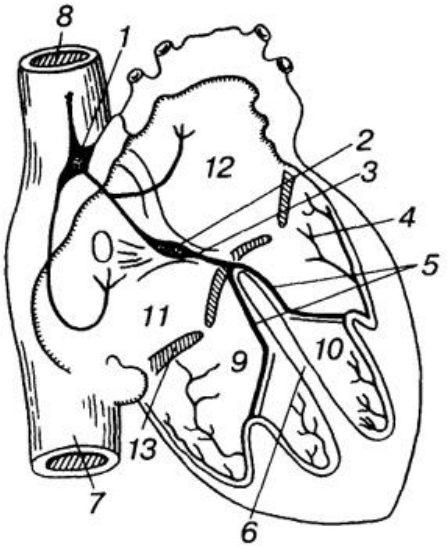


Рисунок 1.3 – Провідна система серця:

1 – синоатріальний вузол; 2 – атріовентрикулярний вузол; 3 – передсердно-шлуночковий пучок; 4 – мережа волокон ніжок передсердно-шлуночкового пучка в стінці шлуночків; 5 – ніжки передсердно-шлуночкового пучка; 6 – міжшлуночкова перегородка; 7 – нижня порожниста вена; 8 – верхня порожниста вена; 9 – правий і 10 – лівий шлуночки; 11 – праве і 12 – ліве передсердя; 13 – передсердно-шлуночкові клапани.

Від синусового вузла процес збудження поширюється на передсердя по передсердним провідним шляхам. Від передсердь збудження переходить на атріовентрикулярний вузол. Минаючи атріовентрикулярне з’єднання, збудження передається на стовбур пучка Гіса, а далі на його розгалуження. Структура внутрішньошлуночкової провідної системи характеризується значною індивідуальною варіабельністю і наявністю різноманітних зв’язків

між основними розгалуженнями. Від спеціалізованих волокон провідної системи збудження переходить на скорочувальний міокард .

Виникнення електричних потенціалів у серцевому м’язі пов’язане з рухом іонів через клітинну мембрану. Основну роль при цьому відіграють катіони натрію та калію. У стані спокою зовнішня поверхня клітини міокарда має позитивний заряд завдяки переважанню катіонів натрію, а внутрішня поверхня клітинної мембрани заряджена негативно через переважання аніонів (Cl^- , HCO^{3-} тощо) усередині клітини (рисунок 1.4)

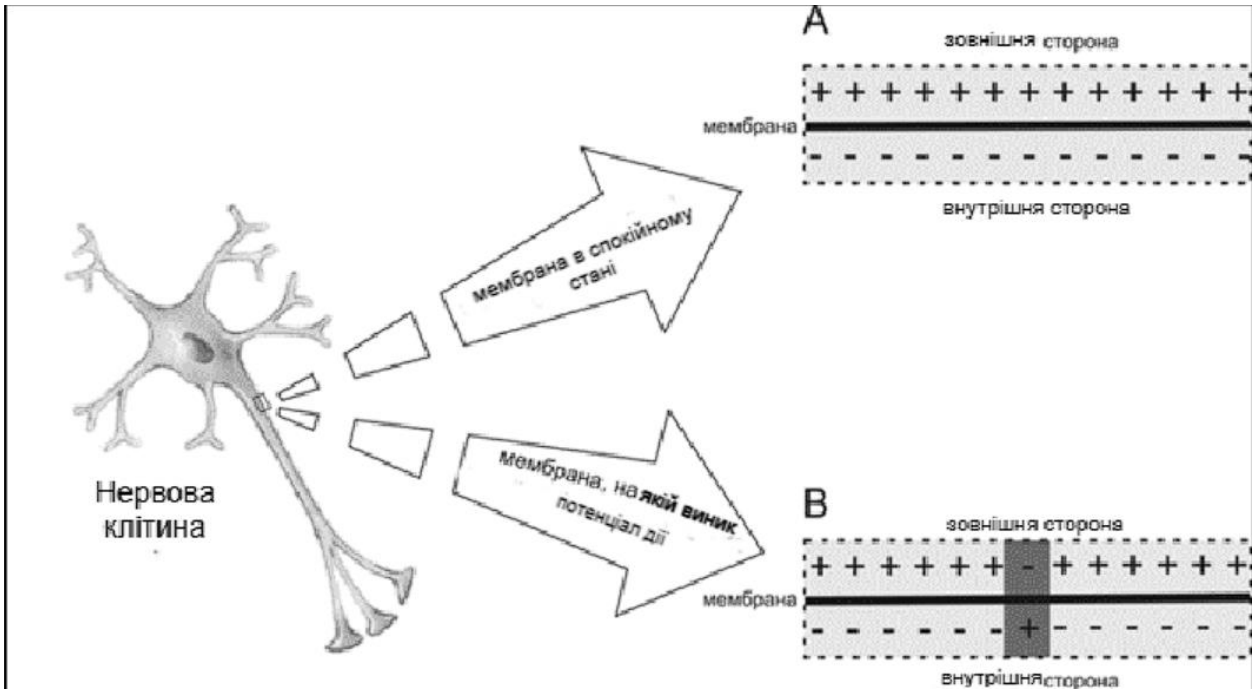


Рисунок 1.4 - Виникнення електричних потенціалів у серцевому м’язі

Зовнішня поверхня даної ділянки набуває негативного заряду через переважання аніонів. При цьому виникає різниця потенціалів між

позитивною і негативною ділянками поверхні клітини, і реєструючий прилад фіксує відхилення від ізоелектричної лінії. Цей процес називається деполяризацією і пов'язаний із потенціалом дії. Незабаром уся зовнішня поверхня клітини набуває негативного заряду, а внутрішня – позитивного, тобто відбувається зворотна поляризація. Зареєстрована крива при цьому повертається до ізоелектричної лінії.

Наприкінці періоду збудження клітинна мембрана стає менш проникною для катіонів натрію, але більш проникною для катіонів калію; останні виходять із клітини (через різницю позаклітинної та внутрішньоклітинної концентрації). Вихід калію з клітини переважає над надходженням натрію в клітину, тому зовнішня поверхня мембрани знову поступово набуває позитивного заряду, а внутрішня – негативного. Цей процес називається реполяризацією. Реєструвальний прилад знову фіксує відхилення кривої, але в інший бік (оскільки полюси клітини змінилися місцями) і меншої амплітуди (оскільки потік іонів калію рухається повільніше).

Описані процеси відбуваються під час систоли. Коли вся зовнішня поверхня знову набуває позитивного заряду, а внутрішня – негативного, буде зафіксована ізоелектрична лінія, що відповідає діастолі. Під час діастоли відбувається повільний зворотний рух іонів калію та натрію, який мало

впливає на заряд клітини, оскільки іони натрію виходять із клітини, а іони калію входять до неї одночасно, і ці процеси врівноважують одне одного.

Описані процеси стосуються збудження окремого волокна міокарда. Імпульс, що виникає при деполяризації, викликає збудження сусідніх ділянок міокарда, поступово охоплюючи весь міокард, розвиваючись за типом ланцюгової реакції .

1.2 Электрокардиография. Системы ЭКГ отведений

Серед численних методів медичної діагностики провідне місце справедливо займає електрокардіографія (ЕКГ). Цей метод дослідження біоелектричної активності серця є незамінним у діагностиці порушень ритму та провідності, гіпертрофії шлуночків і передсердь, ішемічної хвороби серця, інфаркту міокарда та інших захворювань серця.

Електрокардіограма являє собою запис сумарного електричного потенціалу, що виникає під час збудження великої кількості клітин міокарда. ЕКГ записують за допомогою електрокардіографа. Електроди для запису

ЕКГ розміщують на різних ділянках тіла. Система розташування електродів називається електрокардіографічними відведеннями.

Будь-яка ЕКГ складається із зубців, сегментів і інтервалів, що відображають складний процес поширення хвилі збудження серцем. Крива ЕКГ має характерну форму, а піки та інтервали між ними позначаються латинськими літерами P, Q, R, S, T та U (рисунок 1.5).

Для отримання ЕКГ сигналів у системах діагностики та моніторингу використовується безліч різних систем відведень .

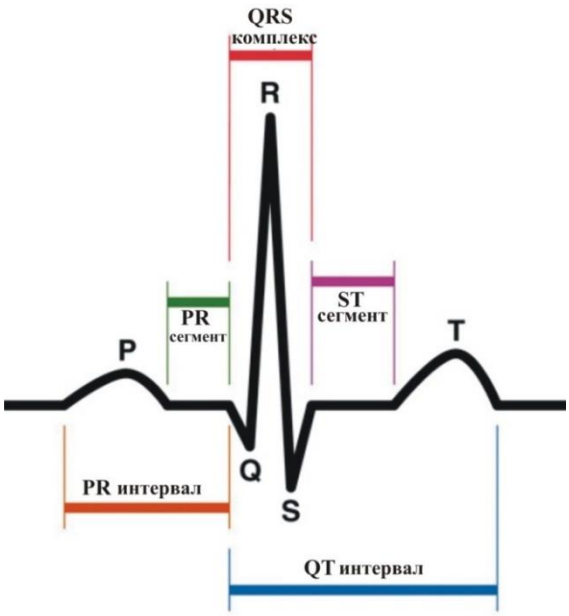


Рисунок 1.5 – Форма сигналу ЕКГ

У клінічній практиці найширше використовують 12 відведень ЕКГ, запис яких є обов'язковим під час обстеження пацієнта. Це 3 стандартних відведення (рисунок 1.6), 3 посилених однополюсних відведення від кінцівок

та 6 грудних відведень. Для формування трьох посилених однополюсних відведень, у якості негативного електрода використовують об'єднаний електрод Голдберга, який утворюється при з'єднанні двох кінцівок через додатковий опір.

Рисунок 1.6 – Двополюсні (стандартні) відведення електрокардіограми (I, II, III - відведення)

Посилені униполярні відведення – aVR, aVL і aVF (рисунок 1.7) також представляють собою вектори у фронтальній площині. Вони вважаються біполярними, оскільки відображають різницю потенціалів між однією точкою та середнім потенціалом двох інших точок зйому.

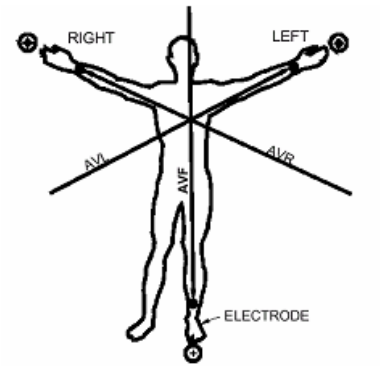


Рисунок 1.7 – Посилені уніполярні відведення

При грудних відведеннях реєструють різницю потенціалів між позитивним електродом, встановленим на поверхні грудної клітки, та негативним об'єднаним електродом Уїльсона. Цей електрод утворюється при з'єднанні через додаткові опори трьох кінцівок (правої руки, лівої руки та лівої ноги), об'єднаний потенціал яких близький до 0 (приблизно 0,2 мВ). Потенціали грудних відведень позначаються великими літерами V1...V6.

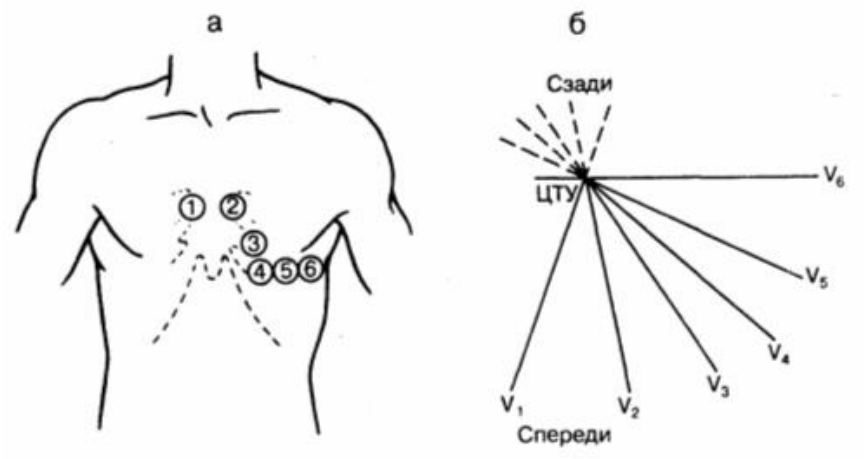


Рисунок 1.6 – Уніполярні грудні відведення (а - розташування грудних електродів від V1 до V6; б - взаємозв'язки між центральною клемою Вільсона і грудним електродом у горизонтальній площині).

Електроди основних відведень накладаються за загальноприйнятою методикою: червоний – R, жовтий – L, зелений – F і чорний – N електроди накладаються відповідно на внутрішню поверхню правого та лівого передпліччя і нижню третину лівої та правої гомілки. За необхідності, наприклад, при проведенні ортостатичної проби, електроди можуть бути

закріплені інакше: попарно на руках (на правому передпліччі – червоний і чорний, на лівому – жовтий і зелений) або ж усі електроди на грудях (відведення за Небом або за Масоном-Лікаром). Переконавшись у стійкій та якісній реєстрації ЕКГ, приступають до її запису в пам'ять комп'ютера.

Відомі й інші системи відведень, які використовуються для отримання ЕКГ в додаткових площинах. Однак використання цих відведень потребує участі медичного персоналу, що має відповідну підготовку. Тому в домашніх кардіографах реалізуються тільки стандартні відведення за Ейнтговеном і Голдбергом.

Таким чином, у роботі буду орієнтуватися на 6-осьову систему за Бейлі .

1.3 Методики виділення QRS - комплексів

На ЕКГ серцевий цикл зазвичай відображається у вигляді трьох легко розрізняних елементів – комплексів. Р-комплекс відповідає деполяризації передсердь, QRS – деполяризації шлуночків, Т – їх реполяризації; реполяризація передсердь на ЕКГ не проявляється. Кожен комплекс складається з кількох різнонаправлених піків, або зубців (рисунок 1.7). Кількість зубців у кожному комплексі різна в різних відведеннях і у різних пацієнтів. Так, R- і Т-комплекси зазвичай містять один або два зубці, а QRS-комплекс – від 1 до 7.

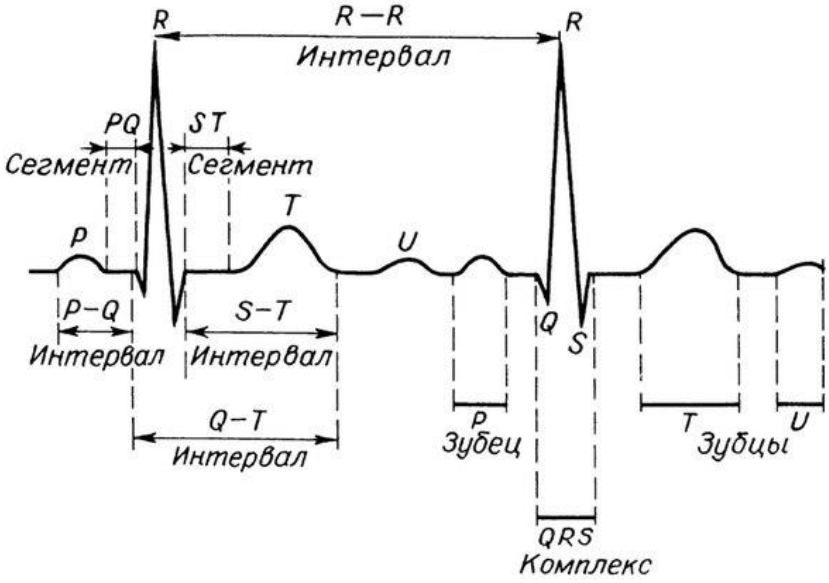


Рисунок 1.7 – Структура серцевого циклу

Тривалість QRS-комплексу відображає час, необхідний для деполяризації м'язів шлуночків, і вимірюється від початку Q-хвилі до кінця S-хвилі. Початком Q-хвилі вважається точка, де рівень ЕКГ-сигналу стає нижчим за рівень ізолінії, а кінцем S-хвилі - точка виходу сигналу на рівень ізолінії, що в значній мірі є умовним. Нормальна тривалість QRS-комплексу складає 0,07-0,10 с. Якщо тривалість перевищує 0,11 с, це може свідчити про наявність порушень провідних шляхів. Максимальна амплітуда QRS-комплексу визначається амплітудою R-хвилі і зазвичай складає 0,5 - 1,0 мВ. Амплітуда QRS-комплексу менше 0,4 мВ може свідчити про певні відхилення.

На даний час існує безліч методик виділення QRS-комплексів, які за використовуваним підходом можна поділити на 5 основних груп:

А. Алгоритми аналізу ЕКГ у часовій області.

Б. Алгоритми, що базуються на частотно-часових, включаючи нелінійні, перетвореннях сигналу ЕКГ.

В. Алгоритми на основі застосування нейромережевих моделей.

Г. Синтаксичні методи.

Д. Комбіновані алгоритми.

Кількісними характеристиками ефективності QRS-алгоритмів прийнято вважати кілька величин: ймовірність розпізнавання довільного QRS-комплексу, виражену у відсотках (чутливість), ймовірність того, що довільно виділений QRS-комплекс є істинним (передбачуваність), та частоту видачі детектором правильного результату (ефективність). Чутливість визначається кількістю хибно відсіяних QRS-обчислень, передбачуваність залежить від кількості хибно визначених комплексів, а ефективність є інтегральним показником якості методу.

Алгоритми групи А, багато з яких використовують принципи, закладені в роботах [14, 18], засновані на застосуванні до вхідного сигналу, крім процедур лінійної фільтрації, зазвичай фільтрів високої чистоти (ФВЧ) і фільтрів низької чистоти (ФНЧ) послідовно, деякого нелінійного перетворення, яке зазвичай включає процедуру інтегрування сигналу в ковзаючому вікні. Положення R-зубця визначається за допомогою порогового детектора рівня сигналу, значення якого можуть бути як фіксованими, так і обчислюватися адаптивно на кожному етапі роботи

алгоритму. Додатково, для підвищення чутливості можуть використовуватися інші процедури підвищення точності: додаткова попередня обробка ЕКГ, введення допоміжних оцінювальних процедур, які дозволяють виявляти ложні або неопределілені QRS-комплекси. Альтернативним підходом є застосування узгодженої фільтрації , що представляє собою обчислення тих чи інших кореляційних співвідношень між відрізком вхідного сигналу відповідної тривалості та базовими шаблонами QRS-комплекса. Цей підхід передбачає різноманітні допоміжні процедури, які дозволяють коригувати морфологію початкових шаблонів. Ефективність методів цієї групи складає 96-98 % правильно визначених QRS-комплексів від загальної кількості, присутніх в ЕКГ.

Алгоритми групи Б засновані на застосуванні після ряду процедур попередньої обробки ЕКГ різних частотно-часових перетворень, таких як локальне перетворення Фур'є, перетворення Карунена-Лоева, дискретне вейвлет-перетворення . При використанні вейвлет-перетворення застосовуються кілька основних базових вейвлет-функцій, причому ряд методів передбачає зворотний зв'язок для корекції їх параметрів. При цьому локалізація положення QRS-комплекса здійснюється в області вейвлет-спектра, у найпростішому випадку шляхом простого детектування рівня. Методи цієї групи характеризуються відносно низькою продуктивністю,

достатньо низькою чутливістю до шумів та ефективністю, що перевищує 99 %.

Алгоритми групи В використовують нейронетеві методи обробки даних і зазвичай застосовуються для аналізу морфології та класифікації елементів ЕКГ. Нейронетеві моделі дозволяють значно ефективніше адаптуватися до нестабільного характеру ЕКГ, тому в задачах виділення QRS-комплексів використовуються при адаптивній узгодженій фільтрації . Чутливість методик варіюється в широкому діапазоні, але в цілому досягає 99 %.

Група Г складається з синтаксичних алгоритмів, також відомих як лінгвістичні або граматичні. Початковий аналізований сигнал представляється у вигляді певної послідовності примітивів, визначаються спеціальні правила (граматики), які породжують той чи інший елемент ЕКГ з множини примітивів. Виділення елементів ЕКГ зводиться до визначення породжуючої граматики .

Група Д складається з різних комбінацій методів, зазвичай являючи собою синтез алгоритмів груп В і Б або В і А. Особливо вигідним виявилось перше поєднання, оскільки такий підхід дозволяє досягти максимальної на сьогоднішній день чутливості — 99,9 % на тестових підбірках ЕКГ. До недоліків методів цієї групи слід віднести вимогливість до обчислювальних ресурсів.

Проведений аналіз показав, що методики груп Б, В, Г і Д вимагають на порядок більше елементарних математичних операцій за одиницю часу, ніж методи групи А, при цьому чутливість і ефективність QRS-детектора зазвичай підвищуються на 1-3 %. Крім того, для адаптації параметрів детектора необхідний більш тривалий інтервал часу, що робить такі алгоритми важко застосовуваними для систем реального часу, коли потрібно отримання ЧСС з перших секунд після початку зняття ЕКГ. Вейвлет-аналізатори зазвичай використовуються при автоматизованому аналізі вже записаної тривалої ЕКГ і відрізняються підвищеною стійкістю до шумів, нейронетеві алгоритми активно застосовуються для класифікації елементів ЕКГ.

Таким чином, я зупинив свій вибір на алгоритмах групи А, оскільки вони показали, що є універсальними, стійкими, мають помірні вимоги до обчислювальних ресурсів і високу ефективність виділення складових навіть при зашумлених сигналах. Крім того, ця група є широко розповсюдженою і легко сумісною з іншими методами аналізу.

2 АНАЛІЗ МЕТОДІВ ОБРОБКИ ТА СЕГМЕНТАЦІЇ ЕКГ- СИГНАЛІВ

2.1 Традиційний алгоритм Пан-Томпкінса

Алгоритм Пан-Томпкінса (Pan-Tompkins) є одним із найвідоміших і широко використовуваних для сегментації ЕКГ-сигналів, зокрема для виявлення QRS-комплексу. Його розроблено для автоматичного аналізу ЕКГ, і він використовує кілька етапів для точного виділення QRS-комплексу.

Зазвичай алгоритм Пан-Томпкінса використовується для виявлення комплексів QRS в реальному часі. Для визначення піків R в комплексах QRS цей підхід використовує нахил, амплітуду та ширину інтегрованого вікна [3]. Алгоритм складається з двох фаз: попередньої обробки та ухвалення рішень. Попередня обробка включає усунення шуму, згладжування сигналу та покращення ширини та нахилу QRS. Сигнал проходить через цифровий фільтр смугового пропускання, який складається з каскадних фільтрів високих і низьких частот. Потім сигнал диференціюється для аналізу інформації про нахил комплексу QRS. Це робиться за допомогою похідної п'яти точок з наступною передавальною функцією:

$$H(z) = (1 / 8T)(-z^{-2} - 2z^{-1} + 2z^{1} + z^{2})$$

У процесі диференціювання низькочастотні хвилі P і T пригнічуються, щоб отримати високочастотні сигнали, що існують у крутіших схилах комплексу QRS. Потім сигнал підноситься до квадрата по точках. Це робить кожну точку даних позитивною та нелінійно підсилює вихід похідної, щоб підкреслити вищі частоти. Це зменшує типово більшу амплітуду хвилі T, яка

може призвести до помилкових виявлень. Застосовується інтеграція рухомим вікном (MWI) для вилучення інформації з особливостей форми хвилі, з урахуванням нахилу хвилі R. Це обчислюється за наступним рівнянням:

$$y(nT) = (1 / N)[x(nT - (N-1)T) + x(nT - (N-2)T) + ... + x(nT)]$$

де, N — це кількість зразків у межах ширини вікна інтеграції.

Визначення ширини рухомого вікна є важливим. Якщо вікно занадто велике, комплекси QRS та T зіллються в інтегрованій хвилі. Якщо воно занадто мале, деякі комплекси QRS можуть генерувати кілька піків в інтегрованій хвилі. Використовується ширина вікна 150 мс, оскільки вона виявилася ефективною.

Фаза ухвалення рішень проводиться для визначення, чи є результат MWI комплексом QRS. Для того, щоб гарантувати, що правильний пік був вибраний, застосовуються два пороги. Перша перевірка сигналу проводиться з використанням вищого з двох порогів (Threshold1). Якщо комплекс QRS не знайдений протягом заздалегідь визначеного проміжку часу, застосовується нижчий поріг (Threshold2), що вимагає застосування методу пошуку назад, щоб знайти комплекс QRS у минулому часі. Пороги коригуються для оптимізації виявлення комплексу QRS. Пороги обчислюються за наступними рівняннями:

$$SPK = 0.125 \ PEAK + 0.875 \ SPK$$

$$NPK = 0.125 \ PEAK + 0.875 \ NPK$$

$$Threshold_1 = NPK + 0.25 \ (SPK - NPK)$$

$$Threshold_2 = 0.5 \ Threshold_1$$

де,

PEAK — це загальний пік

SPK — це поточна оцінка піку сигналу

NPK — це поточна оцінка піку шуму.

У реальних умовах ЕКГ-сигнали часто забруднені різними типами шуму, такими як дрейф основної лінії, перешкоди від електричної мережі, артефакти від м'язів та шуми контактів електродів, що ускладнює проведення морфологічного аналізу . Спектр частот дрейфу основної лінії знаходиться в діапазоні 0,05–1 Гц . Перешкоди від електричної мережі є вузькосмуговим шумом з центром частоти 50/60 Гц. Зазвичай артефакти від м'язів складають близько 10% амплітуди ЕКГ із діапазоном частот 20–1000 Гц. Попередні дослідження показали, що артефакти м'язів можуть значно змінювати форми локальних хвиль, оскільки цей шум часто перекривається з ЕКГ-сигналом у діапазоні частот 0,01–100 Гц [18]. Обмеження: Втрата важливих компонентів сигналу: У алгоритмі Пан-Томпкінса низькочастотні та високочастотні

фільтри з'єднуються для досягнення смуги пропускання близько 5–15 Гц. Однак деякі важливі компоненти ЕКГ-сигналу втрачаються. Присутність випадкового шуму: наявність випадкового шуму погіршує ефективність роботи алгоритму Пан-Томпкінса.

Традиційні методи сегментації ЕКГ, такі як алгоритм Пан-Томпкінса та хвильовий аналіз, мали великий вплив на розвиток автоматизованих систем для обробки ЕКГ-сигналів. Однак їхня ефективність може бути обмежена в умовах сильного шуму або варіацій форми сигналу. З переходом до методів на основі штучних нейронних мереж, можна досягти кращих результатів завдяки здатності глибоких моделей адаптуватися до складних та варіативних даних.

2.2 Сучасні підходи, що базуються на методах машинного навчання, пошук QRS комплексів за допомогою нейронної мережі.

Сучасні підходи до сегментації ЕКГ-сигналів, що базуються на методах машинного навчання, демонструють значні переваги у порівнянні з традиційними методами. Вони дозволяють обробляти складні сигнали, враховувати індивідуальні особливості пацієнтів і адаптуватися до великої варіативності даних.

Рекурентні нейронні мережі мають властивість, що відрізняє їх від інших нейронних мереж. Це мережі з циклами, що дозволяють інформації зберігатися.

На рис. 2.1 частина нейронної мережі має вхід x_t вихід h_t . Цикл дозволяє передавати інформацію від одного кроку мережі до іншого.

Рекурентну нейронну мережу можна представляти як декілька копій однієї і тієї ж мережі, кожна з яких передає повідомлення наступнику.

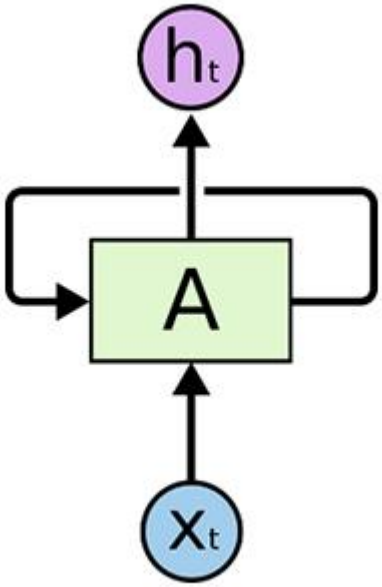


Рисунок 2.1 – Рекурентна мережа

Розглянемо, що відбувається у циклі (рис. 2.2).

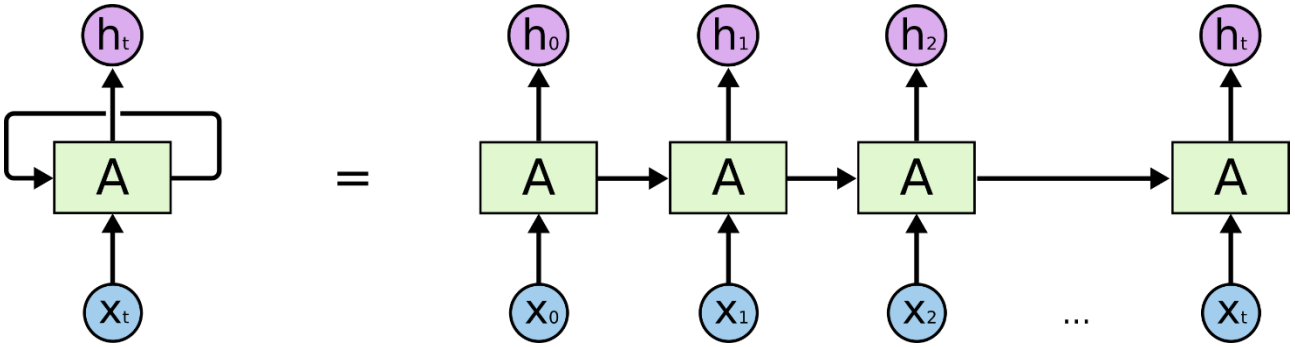


Рисунок 2.1 - Розгорнута рекурентна нейронна мережа

Ланцюгоподібна структура показує, що рекурентні нейронні мережі тісно пов'язані з послідовностями та списками. Це природна архітектура нейронної мережі, яку можна використовувати для розпізнавання мови, моделювання мови, перекладу, підписи до зображень.

LSTM окремих вид рекурентних нейронних мереж, які працюють для багатьох завдань набагато краще, ніж стандартна версія.

Мережі довготривалої короткострокової пам'яті **LSTM**, являють собою особливий вид **RNN**, здатний вивчати довгострокові залежності. Вони були представлені Хохрайтером і Шмідхубером (1997) та розвинені іншими вченим. 1 Вони прекрасно справляються з найрізноманітнішими завданнями і в даний час широко використовуються.

LSTM спеціально розроблені, щоб уникнути проблеми довгострокових залежностей. Запам'ятовування інформації протягом тривалого періоду часу - це їх властивість за замовчуванням.

Усі рекурентні нейронні мережі мають вигляд ланцюжка повторюваних модулів нейронної мережі. У стандартних **RNN** цей повторюваний модуль матиме дуже просту структуру.

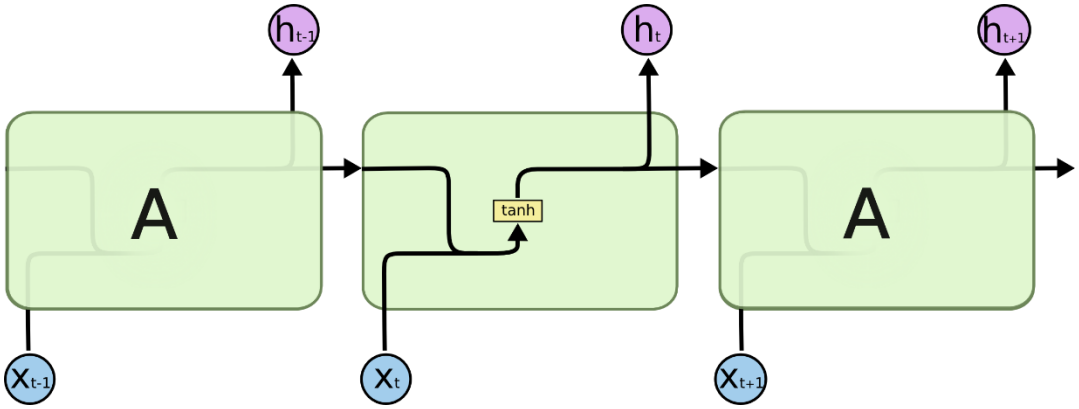


Рисунок 2.2 – Повторювані модулі у стандартній RNN

LSTM також мають таку саму ланцюжкову структуру, але повторюваний модуль має іншу структуру. Замість одного шару нейронної мережі їх чотири, які взаємодіють особливим чином.

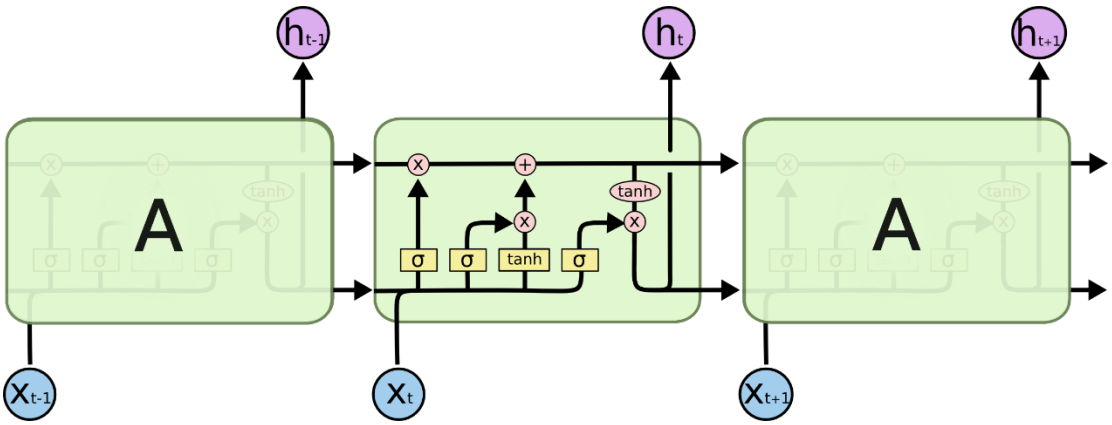


Рисунок 2.3 - Повторювані модулі у LSTM з чотирьома шарами, що взаємодіють

Введемо позначення

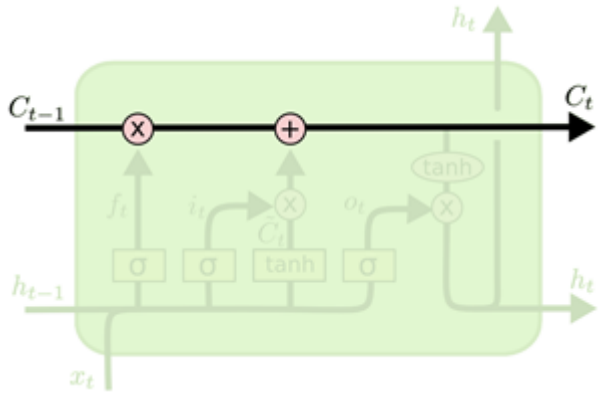
Шар нейронної мережі

Алгебраїчні операції

Вектор

Конкатенація (об’єднання) векторів

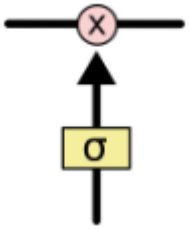
Копіювання векторів та їх розгалуження



Ключовим моментом в LSTM є стан комірки, горизонтальна лінія, що проходить через верхню частину діаграми.

Стан комірки схожий на конвеєрну стрічку. Вона проходить прямо по всьому ланцюжку, лише з деякими незначними лінійними взаємодіями.

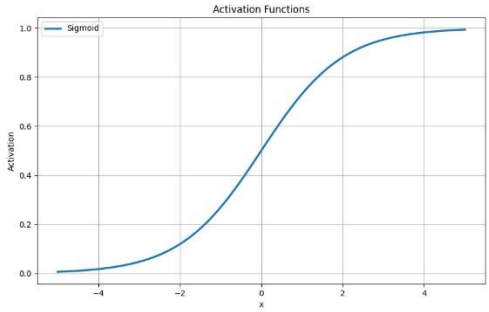
LSTM має можливість видаляти або додавати інформацію до стану пастки, що ретельно регулюється структурами, які називаються ворітьми.



.

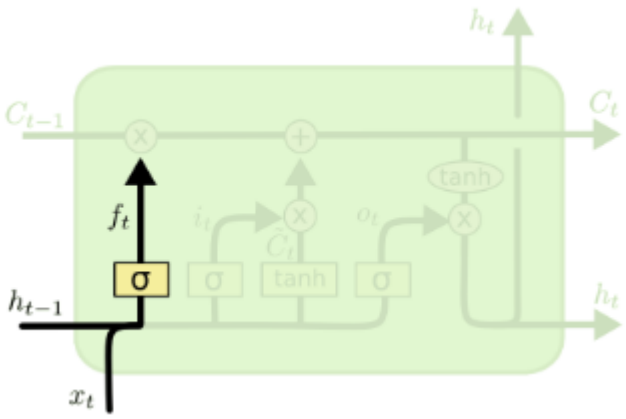
Ворота - це спосіб опціонального пропускання інформації. Вони складаються з сигмоїдного шару нейронної мережі та поелементного помноження.

Сигмоїдна функція $y(s) = \frac{1}{1+\exp(-x)}$ виводить числа від нуля до одиниці, що описують, скільки кожного компонента має бути передано. Значення нуль означає "нічого не пропускати", а значення одиниці означає "пропускати все!"



LSTM має три таких шлюзи для захисту та контролю стану комірки.

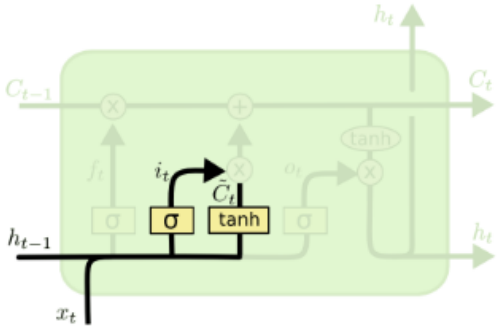
Першим кроком є прийняття рішення про те, якою інформацією можно позбавитись. Це рішення приймається сигмоїдним шаром, який називається "шаром забуття". В залежності від h_i, x_i формується значення f_t. 1 - повністю зберегти, 0 - повністю позбутися.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

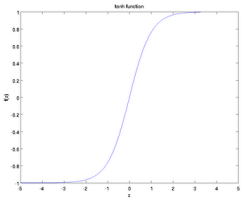
Наступний крок - вирішити, яку нову інформацію необхідно зберігати у стані клітини. Це складається з двох частин. По-перше, сигмоїдний шар, який

називається "шар вхідних воріт", вирішує, які значення ми будемо оновлювати. Далі, тангенціальний шар створює вектор нових значень-кандидатів $\widetilde{C}_t$, які можна додати до стану. На наступному кроці вони об’єднуються та відбивають оновлений стан.



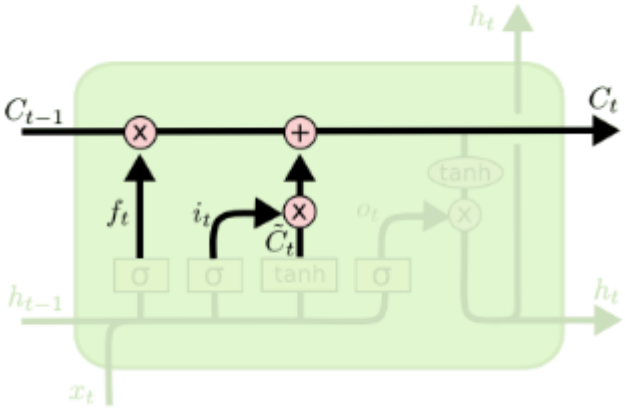
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$



Оновлення стану C_t відбувається з урахуванням коефіцієнтів, які відбивають ступень важливості попереднього стану старого стану C_{t-1} та нового $\widetilde{C}_t$.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t)$$

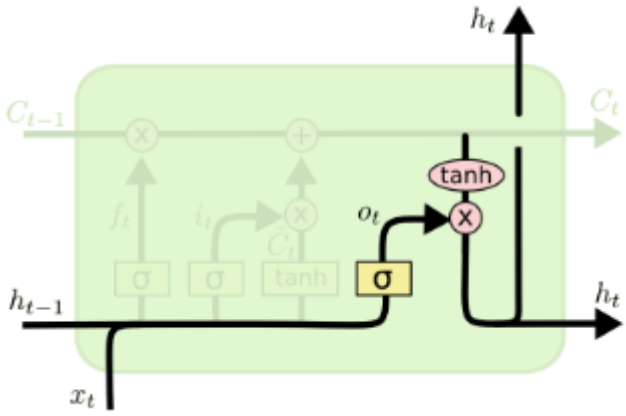


Для визначення вихідного сигналу з осередку LSTM використовується два основні кроки.

Спочатку, стан комірки проходить через шар із сигмоїдною функцією активації. Цей шар визначає, які частини стану будуть виходити назовні, приймаючи значення між 0 і 1.

Потім застосовується гіперболічна функція $f(x) = \frac{2}{1+\exp(-2x)} - 1$ до стану осередку, щоб обмежити значення в діапазоні від -1 до 1. Це допомагає керувати великими і малими значеннями в стані.

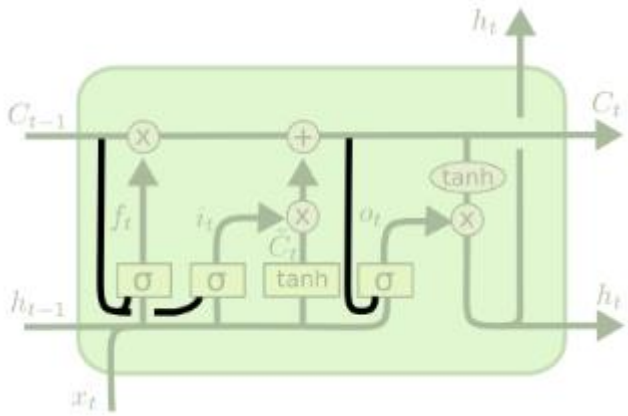
Нарешті, ми множимо значення після гіперболічного тангенса на вихідні значення сигмоїдного шару. Це визначає, які частини стану будуть враховуватися у вихідному сигналі: близькі до 0 значення будуть придушені, а близькі до 1 будуть посилені і включені у вихідний сигнал.



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot tanh(C_t)$$

Існує багато варіацій LSTM. Найбільш популярною є модель, запропонована Герсом і Шмідхубером (2000), яка додає "прорізних з'єднань". Це означає, що ми дозволяємо воротам відстежувати стан комірки.

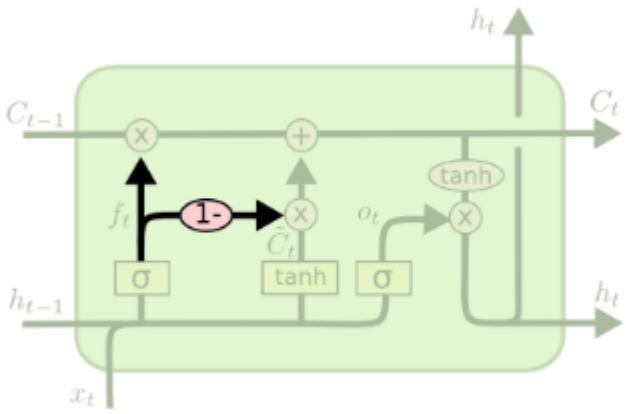


$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$

Інша модифікація полягає в використанні зв'язаних воріт забуття та введення. Замість того, щоб окремо вирішувати, що слід забути і що слід додати нову інформацію, ми робимо ці рішення одночасно. Ми забуваємо лише тоді, коли маємо намір ввести на їх місце щось нове. Нові значення вводяться до стану лише тоді, коли забуваємо щось старе.



$$C_t = f_t \cdot C_{t-1} + (1 - f_t) \cdot \tilde{C}_t)$$

2.3 Порядок виконання практичної роботи

Створення виборки для навчання

Робота базується на результатах пошуку QRS-комплексів методом Пана-Томкінса.

Відкрийте m-file та додайте код, щоб сформувати навчальну вибірку (рис. 2.5). Якщо результати пошуку QRS мають проблемні ділянки (в наданому коді довжина проблемної ділянки дорівнює m відліків), їх необхідно вилучити. На рис. 2.5 показаний початковий фрагмент з некоректним початковим сегментом. Навчальна виборка може бути складеною із декількох діапазонів.

В рядку 135 створюється нульвий вектор **qrs**. Його значення потім замінюються 1 для знайдених QRS-відліків (рядок 136). Амплітуди відповідних відліків заносяться до масива **xTain** (рядок 137). Далі створюється цільовий вектор **qrsTrain** (рядок 138). Навчальна вибірка зберігається в окремому файлі ‘train.mat’ зі структурою двох змінних: ‘**xTrain’**, ‘**qrsTrain’**.

```
133      %% Вибірка для навчання
134 -    m=600;
135 -    qrs(1:length(indR))=0;
136 -    qrs(Rcand)=1;
137 -    xTrain=x(m:end);
138 -    qrsTrain=qrs(m:end);
139 -    save('trainECG.mat','xTrain','qrsTrain');
```

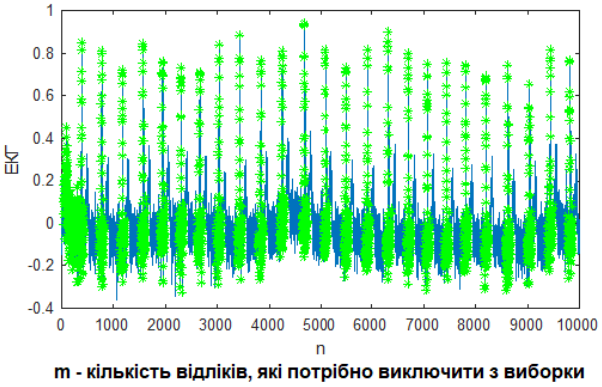


Рисунок 2.4 – Створення навчальної виборки

Підготовка виборки до навчання

Перед застосуванням виборку необхідно перетворити таким чином, щоб вона містила зразки з окремими QRS-комплексами та ділянки, що не містять комплексів.

Для того, щоб сформувати такі зразки необхідно враховувати тривалість QRS-комплексів та визначити їх початок та кінець (рис. 2.6).

Рисунок 2.5 – Принцип визначення окремих QRS

Цей принцип реалізований у коді на рис. 2.7. Спочатку знаходимо індекси, в яких значення змінюється з 0 на 1, що відповідає початку QRS комплексу. Потім знаходимо індекси, в яких значення змінюється з 1 на 0, що відповідає кінцю QRS комплексу.

Далі перевіряємо, чи однакові кількості початкових та кінцевих індексів. Якщо кількість відрізняється, видається повідомлення про помилку (рядки 17-19).

Нарешті, створюємо вектор QRS, де кожен елемент це вкладений масив з початковим та кінцевим індексом QRS комплексу (рядки 25-27).

```
7     %% Формування зразків
8     % Розмірність навчальної виборки
9 -   disp(['Розмір X_train: ', num2str(size(f.xTrain)), ', Розмір qrs_train: ', num2str(size(f.qrsTrain))]);
10    % Індекси,значення яких змінюються з 0 на 1 (початок QRS)
11 -  startIdx = find(diff([0, f.qrsTrain]) == 1);
12
13    % Індекси,значення яких змінюються з 1 на 0 (кінець QRS)
14 -  endIdx = find(diff(f.qrsTrain) == -1);
15
16    % Перевіряем відпвідність кількості початкових та кінцевих індексів
17 -  if numel(startIdx) ~= numel(endIdx)
18 -      error('Помилка: Не вдалося знайти початок або кнінець QRS');
19 -  end
20
21    % Створюємо вектор з вкладеними масивами
22 -  QRS = cell(1, numel(startIdx));
23
24    % Зберігаємо індекси початку та кінця QRS
25 -  for i = 1:numel(startIdx)
26 -      QRS{i} = {(startIdx(i)), (endIdx(i))};
27 -  end
```

Рисунок 2.6 – Визначення Напочатку та кінця QRS

Результат фрагменту коду продемонстрований на рис. 2.8.

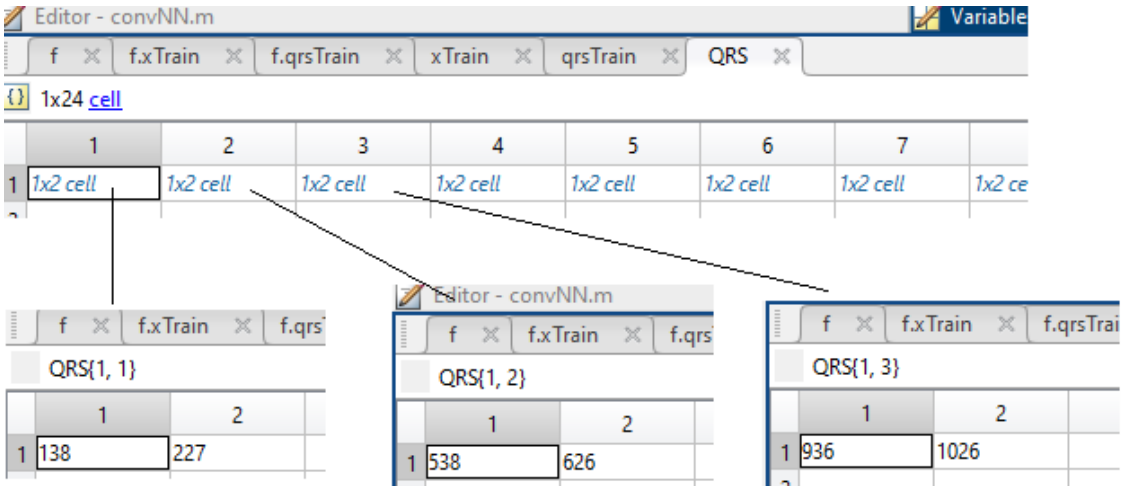


Рисунок 2.7 – Масив QRS

На рис. 2.9 наданий код для формування виборки для нейронної мережі.

Визначимо довжину кожного сегменту даних **numSegm**.

Для кожного індексу **QRS**, зазначеного в масиві **kQRS**, витягують сегмент із навчальних даних **f.xTrain** і зберігають у комірку **segm{i}**. Це виконується шляхом знаходження початкового і кінцевого індексів QRS і вилучення даних із **f.xTrain** у межах цього діапазону. Кожен сегмент містить **numSegm** послідовних відліків.

Для кожного індексу без QRS, зазначеного в масиві **noQRS**, також витягують сегмент із навчальних даних **f.xTrain** і зберігають у комірку **segmNo{i}.**

Отримані сегменти **QRS** і без **QRS** об'єднуються разом у навчальний набір даних **XTrain**, де кожен сегмент представлений як рядок у комірці.

Створюється вектор категоріальних міток **YTrain**, який вказує, чи містить кожен сегмент QRS (мітка 1) чи ні (мітка 0).

Таким чином, **XTrain** містить сегменти даних, а **YTrain** містить відповідні мітки для класифікації QRS і без QRS.

```
%% Навчальна виорки для мережі
%Довжина зразка
numSegm=100;
% Обрані QRS
kQRS=[2,4,6,8,9];
for i=1:numel(kQRS)
    segm{i}=f.xTrain(cell2mat(QRS{1,i}(1)):cell2mat(QRS{1,i}(1))+numSegm);
end

% Обрані зразки без QRS
noQRS=[700,1100,1800,2150];
for i=1:numel(noQRS)
    segmNo{i}=f.xTrain(noQRS(i):noQRS(i)+numSegm);
end

% Объединяем сегменты в обучающую выборку
XTrain = [segm, segmNo]';
YTrain = categorical([1, 1,1, 1, 1, 0, 0,0,0])';
```

Рисунок 2.8 – Формування виборки для мережі

2.4 Створення нейронної мережі на практиці

Для вирішення задачі застосуємо рекурентну нейронну мережу (RNN) з двоспрямованою довготривалою короткостроковою пам'яттю (BiLSTM) для класифікації часових послідовностей.

%% Рекурентна мережа з пам'ятью

```
numObservations = numel(XTrain);
for i=1:numObservations
    sequence = XTrain{i};
    sequenceLengths(i) = size(sequence,2);
end
[sequenceLengths,idx] = sort(sequenceLengths);
XTrain = XTrain(idx);
YTrain = YTrain(idx);
inputSize = 1;
numHiddenUnits = 100;
numClasses = 2;

layers = [ ...
    sequenceInputLayer(inputSize)
    bilstmLayer(numHiddenUnits,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer]
```

```
maxEpochs = 100;

miniBatchSize = 27;


options = trainingOptions('adam', ...

    'ExecutionEnvironment','cpu', ...

    'GradientThreshold',1, ...

    'MaxEpochs',maxEpochs, ...

    'MiniBatchSize',miniBatchSize, ...

    'SequenceLength','longest', ...

    'Shuffle','never', ...

    'Verbose',0, ...

    'Plots','training-progress');


net = trainNetwork(XTrain,YTrain,layers,options);
```

Визначається кількість навчальних прикладів у **XTrain**, і для кожного прикладу визначається його довжина в кількості відліків, зберігаючись у **sequenceLengths**.

Довжини послідовностей сортуються в порядку зростання, а також зберігаються відповідні індекси відсортованих послідовностей.

Навчальні дані **XTrain** і мітки **YTrain** перевпорядковуються відповідно до відсортованих індексів.

Визначаються розмір вхідних даних(**inputSize**), кількість прихованих блоків(**numHiddenUnits**) і кількість класів(**numClasses**). Створюються шари нейронної мережі, включно з вхідним шаром для послідовних даних, двоспрямованим шаром **LSTM**, повнозв'язаним шаром і шаром активації **softmax** для класифікації, а також шаром класифікації. Задаються параметри навчання: алгоритм оптимізації (адаптивний метод оптимізації **Adam**), кількість епох навчання, розмір міні-пакету та інші параметри. Мережа навчається на навчальних даних **XTrain** з відповідними мітками **YTrain** з використанням заданих шарів і параметрів навчання.

2.5 Тестування мережі

```
%% Тестування

segment5=f.xTrain(3400:3400+numSegm);

segment6=f.xTrain(7980:7980+numSegm);

segment7=f.xTrain(8798:8799+numSegm);
```

```
XTest = {segment5,segment6,segment7}';

YTest=categorical([0, 1,1])';

numObservationsTest = numel(XTest);

for i=1:numObservationsTest

    sequence = XTest{i};

    sequenceLengthsTest(i) = size(sequence,2);

end

[sequenceLengthsTest,idx] = sort(sequenceLengthsTest);

XTest = XTest(idx);

YTest = YTest(idx);

YPred = classify(net,XTest, ...

    'MiniBatchSize',miniBatchSize, ...

    'SequenceLength','longest');

analyzeNetwork(layers);

analyzeNetwork(net.Layers);
```

В наведеному вище коді визначається три сегменти даних **segment5**, **segment6** і **segment7**, які будуть використовуватися для тестування. Вони

об'єднуються в масив **XTest** і є вхідними даними мережі. Мітки класів для тестових даних у змінній **YTest** (на приналежність до класу кодується 0 або 1). Далі визначається довжина зразків з **XTest**. Довжини послідовностей сортуються в порядку зростання, і тестові дані перевпорядковуються відповідно до цього порядку. На тестових даних **XTest** викликається метод **classify**, який використовує навчену нейронну мережу **net** для прогнозування класів (результат зберігається у векторі **YPred**).

Після тестування викликаються функції **analyzeNetwork** для виведення параметрів мережі.

3 РОЗРОБКА ПРАКТИЧНОЇ МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ З МАШИННИМ НАВЧАННЯМ ЗА ДОПОМОГОЮ PYTHON

```
  import io
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import LSTM, Dense, Bidirectional,
TimeDistributed
from tensorflow.keras.models import Sequential
import pickle


# Файл з даними для навчальної виборки
file_path = 'train_data.pkl'

# Відкриття файлу
with open(file_path, 'rb') as file:
    loaded_data = pickle.load(file)
```

```
# Зчитування структури
xTrain = loaded_data['xTrain']
qrsTrain = loaded_data['qrsTrain']

#Пошук індексів початку та кінця QRS
start_idx = np.where(np.diff(np.hstack((0, qrsTrain))) == 1)[0]
end_idx = np.where(np.diff(qrsTrain) == -1)[0]

#Перевірка відпвідності знайденої кількості початкових та кінцевих
індексів QRS
if len(start_idx) != len(end_idx):
    raise ValueError('Помилка: Не вдалося знайти початок або кінець QRS')

#Формування масиву, кожен елемент якого вказує на початок та кінець QRS
QRS = []

for i in range(len(start_idx)):
    QRS.append((start_idx[i], end_idx[i]))

# Формування сегментів для навчання

# Довжина сенменту QRS
numSegm = 100

# Номери QRS
kQRS = [2,4,6,8,9]

segm = []
for i in range(len(kQRS)):
    idx = QRS[kQRS[i] - 1]
    segm.append(xTrain[idx[0]:idx[0] + numSegm])

# Довжина сенменту без QRS
numSegm1 = 100

# Номери індексів початку фрагментів
noQRS = [700, 1100, 1800, 2150]
segmNo = []

for i in range(len(noQRS)):
    segmNo.append(xTrain[noQRS[i]:noQRS[i] + numSegm1])

# Навчальна вибірка
XTrain = segm + segmNo

# Масив міток
YTrain = [1, 1, 1, 1, 1, 0, 0, 0, 0]
```

```
# Перетворення в массивы NumPy
XTrain_array = np.array(XTrain)
YTrain_array = np.array(YTrain)

# Параметри моделі і навчання
numHiddenUnits = 100
inputSize = 1
numClasses = 2
maxEpochs = 100
miniBatchSize = 50

# Формаування для мережі LSTM
XTrain_array = XTrain_array.reshape(XTrain_array.shape[0],
XTrain_array.shape[1], 1)

# Побудова моделі мережі
model = Sequential([
    Bidirectional(LSTM(numHiddenUnits, return_sequences=False),
input_shape=(None, inputSize)),
    Dense(numClasses, activation='softmax')
])

# Компіляція моделі
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Навчання мережі
model.fit(XTrain_array, YTrain_array, epochs=maxEpochs,
batch_size=miniBatchSize, verbose=1)
```

Результати

```
Epoch 1/100

1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - accuracy: 0.8889 -
loss: 0.6896
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 0.7778 -
loss: 0.6827
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.7778 -
loss: 0.6759
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 0.6667 -
loss: 0.6689
Epoch 5/100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 0.6667 -
loss: 0.6617
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 0.6667 -
loss: 0.6540
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step - accuracy: 0.6667 -
loss: 0.6459
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step - accuracy: 0.6667 -
loss: 0.6370
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step - accuracy: 0.6667 -
loss: 0.6274
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - accuracy: 0.6667 -
loss: 0.6168
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 0.6667 -
loss: 0.6050
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.7778 -
loss: 0.5911
Epoch 13/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 0.7778 -
loss: 0.5730
Epoch 14/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - accuracy: 0.7778 -
loss: 0.5460
Epoch 15/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - accuracy: 0.5556 -
loss: 0.5101
Epoch 16/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.5556 -
loss: 0.4760
Epoch 17/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - accuracy: 1.0000 -
loss: 0.4369
Epoch 18/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step - accuracy: 1.0000 -
loss: 0.3897
Epoch 19/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 1.0000 -
loss: 0.3202
Epoch 20/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - accuracy: 1.0000 -
loss: 0.2474
Epoch 21/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - accuracy: 1.0000 -
loss: 0.1630
Epoch 22/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - accuracy: 1.0000 -
loss: 0.0736
Epoch 23/100
```

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 141ms/step - accuracy: 1.0000 - loss: 0.0769
Epoch 24/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 152ms/step - accuracy: 1.0000 - loss: 0.0341
Epoch 25/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 129ms/step - accuracy: 1.0000 - loss: 0.0538
Epoch 26/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 144ms/step - accuracy: 1.0000 - loss: 0.1144
Epoch 27/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 139ms/step - accuracy: 1.0000 - loss: 0.0961
Epoch 28/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 143ms/step - accuracy: 0.8889 - loss: 0.1224
Epoch 29/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 130ms/step - accuracy: 1.0000 - loss: 0.0354
Epoch 30/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 133ms/step - accuracy: 1.0000 - loss: 0.0573
Epoch 31/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 137ms/step - accuracy: 1.0000 - loss: 0.0532
Epoch 32/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 138ms/step - accuracy: 1.0000 - loss: 0.0270
Epoch 33/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 154ms/step - accuracy: 1.0000 - loss: 0.0176
Epoch 34/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 125ms/step - accuracy: 1.0000 - loss: 0.0142
Epoch 35/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 149ms/step - accuracy: 1.0000 - loss: 0.0139
Epoch 36/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 131ms/step - accuracy: 1.0000 - loss: 0.0183
Epoch 37/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 162ms/step - accuracy: 1.0000 - loss: 0.0318
Epoch 38/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 276ms/step - accuracy: 1.0000 - loss: 0.0329
Epoch 39/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 142ms/step - accuracy: 1.0000 - loss: 0.0180
Epoch 40/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 300ms/step - accuracy: 1.0000 - loss: 0.0118
Epoch 41/100

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 101ms/step - accuracy: 1.0000 -
loss: 0.0094
Epoch 42/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 143ms/step - accuracy: 1.0000 -
loss: 0.0084
Epoch 43/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 123ms/step - accuracy: 1.0000 -
loss: 0.0080
Epoch 44/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 138ms/step - accuracy: 1.0000 -
loss: 0.0079
Epoch 45/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 148ms/step - accuracy: 1.0000 -
loss: 0.0080
Epoch 46/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 127ms/step - accuracy: 1.0000 -
loss: 0.0080
Epoch 47/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 82ms/step - accuracy: 1.0000 -
loss: 0.0080
Epoch 48/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 79ms/step - accuracy: 1.0000 -
loss: 0.0078
Epoch 49/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 139ms/step - accuracy: 1.0000 -
loss: 0.0073
Epoch 50/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 150ms/step - accuracy: 1.0000 -
loss: 0.0066
Epoch 51/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 81ms/step - accuracy: 1.0000 -
loss: 0.0059
Epoch 52/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 140ms/step - accuracy: 1.0000 -
loss: 0.0053
Epoch 53/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 136ms/step - accuracy: 1.0000 -
loss: 0.0048
Epoch 54/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 138ms/step - accuracy: 1.0000 -
loss: 0.0044
Epoch 55/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 131ms/step - accuracy: 1.0000 -
loss: 0.0042
Epoch 56/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 80ms/step - accuracy: 1.0000 -
loss: 0.0041
Epoch 57/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 85ms/step - accuracy: 1.0000 -
loss: 0.0040
Epoch 58/100
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 94ms/step - accuracy: 1.0000 -
loss: 0.0039
Epoch 59/100

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 1.0000 -
loss: 0.0037
Epoch 60/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 1.0000 -
loss: 0.0034
Epoch 61/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - accuracy: 1.0000 -
loss: 0.0030
Epoch 62/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 1.0000 -
loss: 0.0027
Epoch 63/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - accuracy: 1.0000 -
loss: 0.0024
Epoch 64/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 1.0000 -
loss: 0.0022
Epoch 65/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step - accuracy: 1.0000 -
loss: 0.0020
Epoch 66/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 1.0000 -
loss: 0.0019
Epoch 67/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - accuracy: 1.0000 -
loss: 0.0018
Epoch 68/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - accuracy: 1.0000 -
loss: 0.0017
Epoch 69/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 1.0000 -
loss: 0.0016
Epoch 70/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - accuracy: 1.0000 -
loss: 0.0016
Epoch 71/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - accuracy: 1.0000 -
loss: 0.0015
Epoch 72/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 1.0000 -
loss: 0.0015
Epoch 73/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 1.0000 -
loss: 0.0015
Epoch 74/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 1.0000 -
loss: 0.0014
Epoch 75/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - accuracy: 1.0000 -
loss: 0.0014
Epoch 76/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - accuracy: 1.0000 -
loss: 0.0014
Epoch 77/100
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step - accuracy: 1.0000 -
loss: 0.0013
Epoch 78/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 1.0000 -
loss: 0.0013
Epoch 79/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 1.0000 -
loss: 0.0013
Epoch 80/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 1.0000 -
loss: 0.0013
Epoch 81/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - accuracy: 1.0000 -
loss: 0.0012
Epoch 82/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 1.0000 -
loss: 0.0012
Epoch 83/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - accuracy: 1.0000 -
loss: 0.0012
Epoch 84/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 1.0000 -
loss: 0.0011
Epoch 85/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step - accuracy: 1.0000 -
loss: 0.0011
Epoch 86/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step - accuracy: 1.0000 -
loss: 0.0011
Epoch 87/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step - accuracy: 1.0000 -
loss: 0.0011
Epoch 88/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 1.0000 -
loss: 0.0011
Epoch 89/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 1.0000 -
loss: 0.0010
Epoch 90/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 1.0000 -
loss: 0.0010
Epoch 91/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 1.0000 -
loss: 0.0010
Epoch 92/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 1.0000 -
loss: 9.9121e-04
Epoch 93/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - accuracy: 1.0000 -
loss: 9.7553e-04
Epoch 94/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 148ms/step - accuracy: 1.0000 -
loss: 9.6055e-04
Epoch 95/100
```

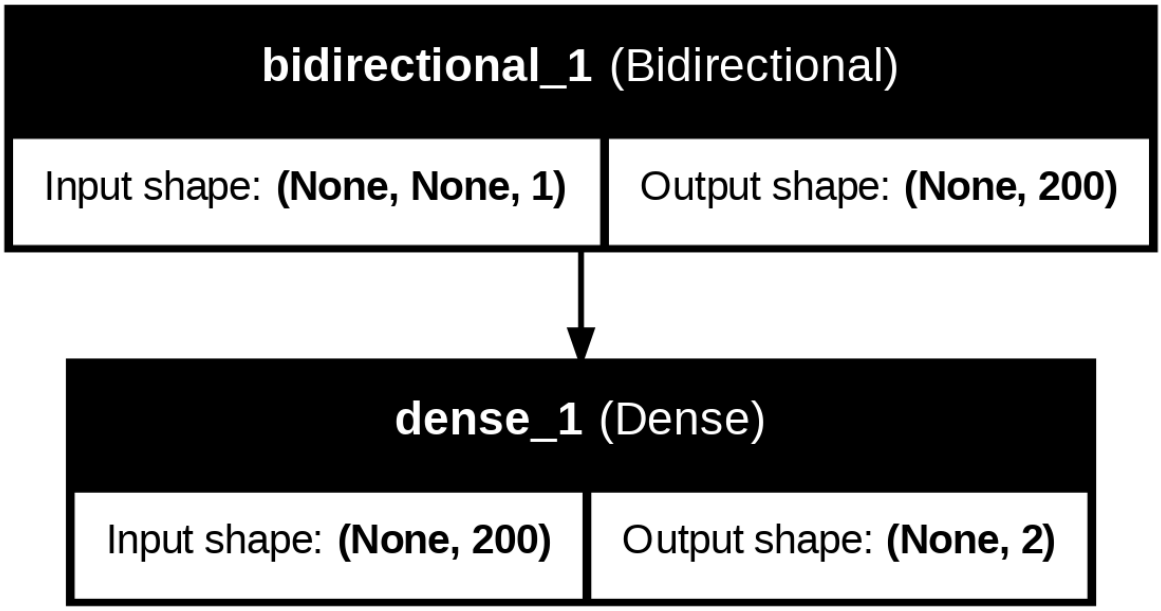
```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 1.0000 -
loss: 9.4622e-04
Epoch 96/100
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - accuracy: 1.0000 -
loss: 9.3252e-04
Epoch 97/100
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - accuracy: 1.0000 -
loss: 9.1934e-04
Epoch 98/100
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 1.0000 -
loss: 9.0670e-04
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - accuracy: 1.0000 -
loss: 8.9451e-04
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 128ms/step - accuracy: 1.0000 -

loss: 8.8275e-04
```

Тестування

```
from tensorflow.keras.utils import plot_model

# Тестування
numSegm2 = 120

# Створення списку для тестових даних
XTest = []
segment1=xTrain[4460:4460 + numSegm2] #  QRS
segment2=xTrain[4870:4870 + numSegm2] #  QRS
segment3=xTrain[5680:5680 + numSegm2] #  QRS
segment4=xTrain[4600:4600 + numSegm2] #  немає QRS
segment5 = xTrain[3400:3400 + numSegm2] # немає QRS
segment6 = xTrain[7980:7980 + numSegm2] #  QRS
segment7 = xTrain[8798:8798 + numSegm2] #  QRS
XTest.append(segment1)
XTest.append(segment2)
XTest.append(segment3)
XTest.append(segment4)
XTest.append(segment5)
XTest.append(segment6)
XTest.append(segment7)

# Перетворення списку в масив NumPy
XTest_array = np.array(XTest)

# Масив міток
YTest = [1,1,1,0,0, 1, 1]
```

```
# Прогноз міток
YPred = np.argmax(model.predict(XTest_array, batch_size=miniBatchSize),
axis=1)

# Обчислення точності
correct_predictions = sum(1 for pred, true in zip(YPred, YTest) if pred ==
true)
total_predictions = len(YTest)
accuracy = correct_predictions / total_predictions
print("Точність:", accuracy)

# Візуалізація архітектури мережі
plot_model(model, to_file='model_architecture.png', show_shapes=True,
show_layer_names=True)
```

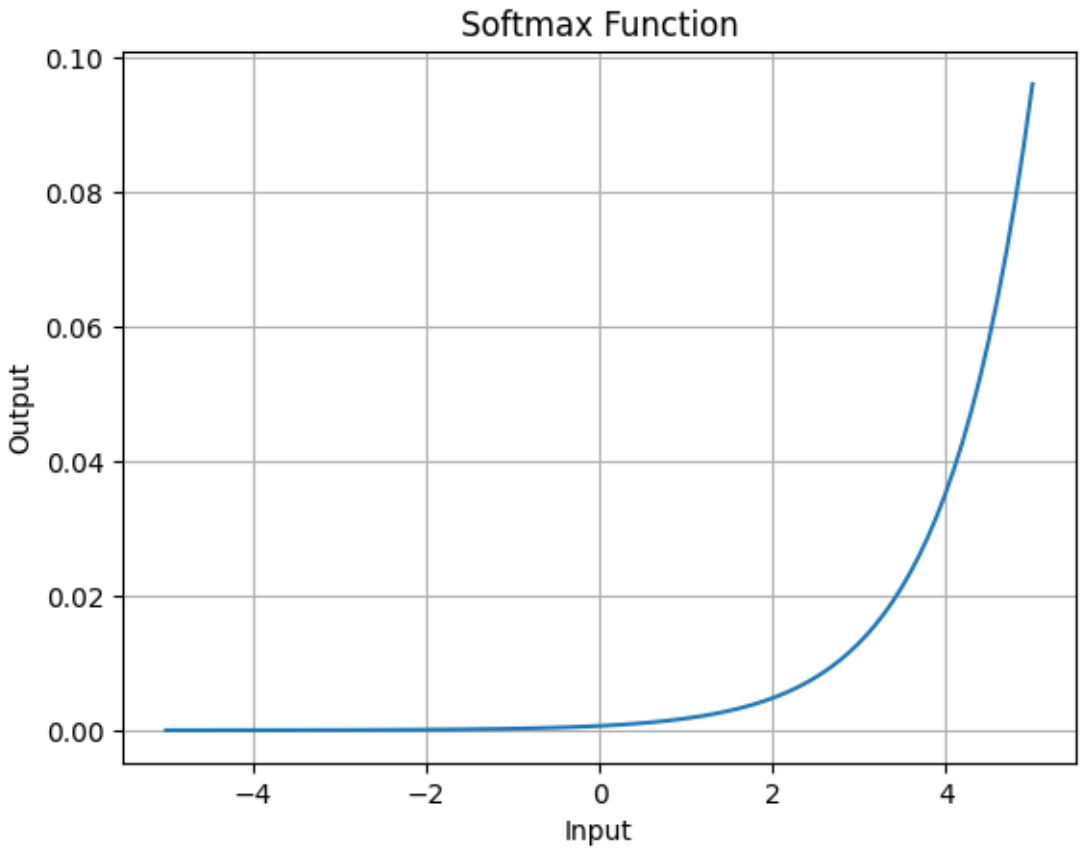
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
Точність: 0.8571428571428571
```




```
numSegm2 = 100
XTest = []
XTest.append(segment5)
XTest.append(segment6)
XTest.append(segment7)
```

```python
# Информация о слоях
print("Layer information:")
for layer in model.layers:
    print("Layer:", layer.name)
    print("Type:", type(layer).__name__)
    if hasattr(layer, 'activation'):
        print("Activation:", layer.activation.__name__)
    if hasattr(layer, 'units'):
        print("Number of units:", layer.units)
    if hasattr(layer, 'output_shape'):
        print("Output shape:", layer.output_shape)
    if hasattr(layer, 'trainable'):
        print("Trainable:", layer.trainable)
    print()

import matplotlib.pyplot as plt
import numpy as np

# Значения по оси x (входы)
x_values = np.linspace(-5, 5, 100)

# Рассчитываем softmax для каждого значения по оси x
softmax_values = np.exp(x_values) / np.sum(np.exp(x_values), axis=0)

# Построение графика
plt.plot(x_values, softmax_values)
plt.title('Softmax Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.grid(True)
plt.show()
```

```
Layer information:
Layer: bidirectional_1
Type: Bidirectional
Trainable: True

Layer: dense_1
Type: Dense
Activation: softmax
Number of units: 2
Trainable: True
```



З кожним циклом навчання, нейронна мережа збільшувала здатність розпізновати QRS – комплекси та підвищувала точність. Переваги RNN для задачі сегментації QRS

1. Моделювання послідовностей і залежностей у часі.

2. Автоматичне навчання ознак, без потреби в ручній розробці фільтрів.

3. Висока точність за умови наявності якісного набору даних.

ВИСНОВКИ

1. У першій частині дипломної роботи проведено медико-технічне обґрунтування дослідження. Розглянуто фізіологічні основи генезу електрокардіограми та відведення для її реєстрації.

2. Аналіз існуючих методів: Проведено огляд традиційних методів сегментації, таких як алгоритм Пан-Томпкінса. Визначено його основні переваги та обмеження, що стало відправною точкою для пошуку більш гнучкого і точного рішення.

3. Сучасні підходи: Розглянуто новітні методи на основі машинного навчання, серед яких особливу увагу приділено рекурентним нейронним мережам (RNN) і їх модифікаціям, таким як LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit), які демонструють високу ефективність у роботі з часовими рядами.

4. Реалізація :

- Розроблено програмну реалізацію сегментації ЕКГ-сигналів із використанням RNN.
- Використано Python та бібліотеки TensorFlow і Keras для побудови, тренування та тестування моделі.
- Проведено попередню обробку сигналів (фільтрацію, нормалізацію) для покращення результатів моделі.

5. Результати моделювання:

- Модель RNN успішно розпізнала ключові компоненти ЕКГ-сигналу (P-хвилі, QRS-комплекс, T-хвилі) з точністю до 85%, перевершуючи деякі традиційні підходи.
- Продемонстровано стабільність моделі на тестових даних із різними рівнями шуму.

6. Переваги використання RNN:

- Ефективне врахування довготривалих залежностей у сигналі.
- Можливість обробки нестаціонарних та зашумлених даних.

7. Практична цінність:

- Розроблена система може бути інтегрована в медичне обладнання для автоматичного аналізу ЕКГ.
- Використання методів машинного навчання дозволяє адаптувати систему до різних фізіологічних умов пацієнтів, що значно підвищує точність діагностики.

У підсумку, результати роботи підтвердили ефективність застосування методів машинного навчання, зокрема RNN, для задач сегментації ЕКГ-сигналів. Подальші дослідження можуть бути спрямовані на інтеграцію гібридних підходів, таких як поєднання CNN і RNN, для покращення точності та швидкості роботи системи.

СПИСОК ДЖЕРЕЛ

1. Статистика сердечно-сосудистых заболеваний в Украине [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: Режим доступа: www/ URL: http://www.provisor.com.ua/archive/2002/N22/art_36.php. - Загл. с экрана.

2. Пан, Дж., Томпкінс, В. Дж. (1985). Алгоритм реального часу для виявлення QRS. IEEE Transactions on Biomedical Engineering, 32(3), 230-236.

3. Голубєв, В. Г., та ін. (2015). Аналіз електрокардіограм із використанням методів цифрової обробки сигналів. Вісник НТУ "ХПІ", № 17, 32-38.

4. Седінкін, В. М., Висоцький, І. С. (2019). Основи аналізу біомедичних сигналів. Київ: Видавництво НТУУ "КПІ".

5. Жуков, В. П. (2020). Застосування методів машинного навчання для обробки сигналів ЕКГ. Вісник Харківського національного університету, № 25, 45-51.

6. Дзюба, О. О., Заярна, Т. В. (2021). Використання Python для аналізу електрокардіографічних даних. Інформаційні технології та комп'ютерна інженерія, 3, 56-61.

7. Хміль, О. І. (2022). Застосування нейронних мереж для аналізу фізіологічних сигналів. Український журнал біомедичних досліджень, 5(2), 18-24.

8. Глушков, В. М., та ін. (2020). Основи цифрової обробки сигналів. Київ: Наукова думка.

9. Goldberger, A. L., et al. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation, 101(23), e215-e220.

10. Martis, R. J., et al. (2013). Application of Principal Component Analysis to ECG Signal Processing. Medical & Biological Engineering & Computing, 51(5), 507-517.

11. Hannun, A. Y., et al. (2019). Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. Nature Medicine, 25, 65-69.

12. Zihlmann, M., Perekrestenko, D., & Tschannen, M. (2017). Convolutional Recurrent Neural Networks for Electrocardiogram Classification. arXiv preprint arXiv:1710.06122.

13. Yang, W., et al. (2018). ECG Signal Classification with Deep Learning. Journal of Signal Processing Systems, 91(3), 283-293.

14. TensorFlow Documentation. Guide to Time Series Forecasting and Signal Analysis. Офіційний сайт.

15. Chollet, F. (2017). Deep Learning with Python. Manning Publications.

16. Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

17. Заводська, А. І., та ін. (2022). Використання LSTM для аналізу біомедичних сигналів. Науковий вісник НУХТ, 4, 36-42.

18. Zabihi, M., et al. (2019). Detection of Myocardial Infarction Using RNN-Based Models. Computers in Biology and Medicine, 113, 103382.

19. Ганенко, А. О. (2021). Дослідження сегментації ЕКГ з використанням глибоких нейронних мереж. Сучасні інформаційні системи, 3, 25-31.

20. Schäfer, A., & Karrenbauer, A. (2021). Transformer-Based Models for Time-Series Analysis in Biomedical Data. Frontiers in Artificial Intelligence, 4, 663890.

21. PhysioNet Database. Publicly Available Database of Annotated ECG Signals.