Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-19-2* _____ *І.І. Федорчук*
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: *док. філ., ст. викладач кафедри КН* _____ *П.М. Радюк*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис — Ініціали, прізвище

05 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*
Освітня програма *освітньо-професійна програма підготовки бакалавра*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак
«06» 03 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі»*

2. Завдання видано студенту *Федорчуку Ігору Ігоровичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *ст. викладач кафедри КН Радюк Павло Михайлович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «01» 03 2023р. № 5

5. Дата видачі завдання студенту: «03» 03 2023р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Провести аналіз методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм та обрати найкращий; застосувати обраний спосіб цифрового опису та формалізації сигналів електрокардіограм до розв'язання задачі автоматизованого окреслення сигналів електрокардіограм; реалізувати обраний спосіб у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм; провести експериментальне тестування реалізованого модуля за еталонними наборами даних; вихідними даними є координати окреслених сигналів електрокардіограм.

'. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*грудень 2022*	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2023*	*виконано*
3	*Робота над розділом 1 – Дослідження процесу окреслення сигналів електрокардіограм засобами штучного інтелекту*	*січень 2023*	*виконано*
4	*Робота над розділом 2 – Спосіб цифрового окреслення сигналів електрокардіограм*	*березень 2023*	*виконано*
5	*Робота над розділом 3 – Програмна реалізація інформаційної системи з використанням обраного способу окреслення сигналів електрокардіограм*	*квітень 2023*	*виконано*
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2023*	*виконано*
7	*Попередній захист кваліфікаційної роботи бакалавра*	*травень 2023*	*виконано*
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	*червень 2023*	*виконано*

Виконавець: *студент 4 курсу, група КН-19-2* (Курс, група виконавця) _____ (Підпис) *І.І. Федорчук* (Ініціали, прізвище)

Керівник: *док. філ., ст. викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *П.М. Радюк* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-2 Федорчук Ігор Ігорович*

Керівник кваліфікаційної роботи бакалавра: *доктор філософії, ст. викладач кафедри КН Радюк Павло Михайлович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
65	24	13	31	4

Метою кваліфікаційної роботи бакалавра є покращення окреслення сигналів електрокардіограм.

Зміст пояснювальної записки та вихідні дані: провести аналіз методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм та обрати найкращий; застосувати обраний спосіб цифрового опису та формалізації сигналів електрокардіограм до розв'язання задачі автоматизованого окреслення сигналів електрокардіограм; реалізувати обраний спосіб у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм; провести експериментальне тестування реалізованого модуля за еталонними наборами даних; вихідними даними є координати окреслених сигналів електрокардіограм.

Досягнення мети кваліфікаційної роботи бакалавра полягає у створенні інформаційної системи, використання якої дасть змогу покращити процес окреслення сигналів електрокардіограм.

Ключові слова: сигнал електрокардіограми, окреслення, згорткова нейронна мережа, автокодувальна нейромережа.

Виконавець: *студент 4 курсу, група КН-19-2* _____ *І.І. Федорчук*

Курс, група виконавця | Підпис | Ініціали, прізвище

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ЕКГ	Електрокардіограми
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
ВСР	Варіабельність серцевого ритму
ШІЗ	Синдром гострого інтоксикаційного захворювання
CNN	Convolutional neural network
CSV	Comma-separated values

Вступ

Кваліфікаційна робота бакалавра присвячена розв’язанню задачі окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі.

Актуальність.

Актуальність використання автокодерів для аналізу сигналів ЕКГ полягає в тому, що вони можуть надати цінну інформацію про стан здоров’я пацієнта. Автокодери можна використовувати для виявлення аритмій, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень. Автокодери також можна використовувати для виявлення аномалій у сигналі, таких як зміни частоти серцевих скорочень або аномальні ритми.

Нейронні мережі автокодерів використовують для аналізу сигналів електрокардіограми завдяки їхній здатності стискати та реконструювати вихідні дані з мінімальною втратою інформації. Автокодери можна використовувати для виділення ознак, класифікації, виявлення аномалій та інших завдань, пов’язаних із моніторингом здоров’я серця. Порівняно з традиційними методами, такими як вейвлет-перетворення або методи аналізу Фур’є, автокодери пропонують кілька переваг, таких як швидший час навчання, краща точність і більша гнучкість щодо того, які характеристики можна витягти із сигналу. Крім того, автокодувальники вимагають менше спеціальних знань порівняно з традиційними методами, що спрощує їх використання.

Використовуючи методи, способи та технології глибокого навчання, як автокодери, можна отримати цінну інформацію про здоров’я серця, яка, можливо, була б неможливою за допомогою лише традиційних методів.

Застосування автокодерів для аналізу сигналів ЕКГ є досить обіцяючим напрямком досліджень у галузі медицини. Їхній потенціал полягає у здатності ефективно виявляти аритмії, ідентифікувати шаблони та прогнозувати майбутні значення, а також виявляти аномалії в сигналі, що дозволяє здійснювати більш точний моніторинг стану здоров'я пацієнта.

У порівнянні з традиційними методами аналізу ЕКГ, автокодери мають декілька переваг, включаючи швидкий час навчання, високу точність та гнучкість у витягуванні корисних характеристик з сигналу. Крім того, використання автокодерів не вимагає великої кількості спеціальних знань, що робить їх доступними та легкими у використанні для медичних фахівців.

Завдяки використанню глибокого навчання та методів штучного інтелекту, автокодери можуть значно покращити аналіз сигналів ЕКГ та надати додаткову інформацію про стан здоров'я серця, що може бути недосяжним за допомогою традиційних підходів. Це відкриває нові перспективи для розвитку урбаністичної медицини та поліпшення діагностики та лікування серцево-судинних захворювань.

Об’єкт дослідження – процес опису, формалізації та окреслення сигналів електрокардіограм.

Предмет дослідження – методи, способи та технології цифрового опису та формалізації сигналів електрокардіограм.

Мета кваліфікаційної роботи бакалавра – покращення окреслення сигналів електрокардіограм.

Завдання кваліфікаційної роботи бакалавра – для досягнення поставленої мети визначено наступні завдання:

1. Провести аналіз методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм та обрати найкращий.

2. Застосувати обраний спосіб цифрового опису та формалізації сигналів електрокардіограм до розв’язання задачі автоматизованого окреслення сигналів електрокардіограм.

3. Реалізувати обраний спосіб у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм.

4. Провести експериментальне тестування реалізованого модуля за еталонними наборами даних.

Розділ 1 Дослідження процесу окреслення сигналів електрокардіограм засобами штучного інтелекту

1.1 Огляд процесу окреслення сигналів електрокардіограм під час медичного діагностування

Електрокардіограма – це медичний тест, який реєструє електричну активність серця. Він використовується для виявлення аномалій серця, діагностики захворювань серця та контролю ефективності лікування [1]. ЕКГ проводиться шляхом розміщення електродів на грудях, руках і ногах (рисунок 1.1).

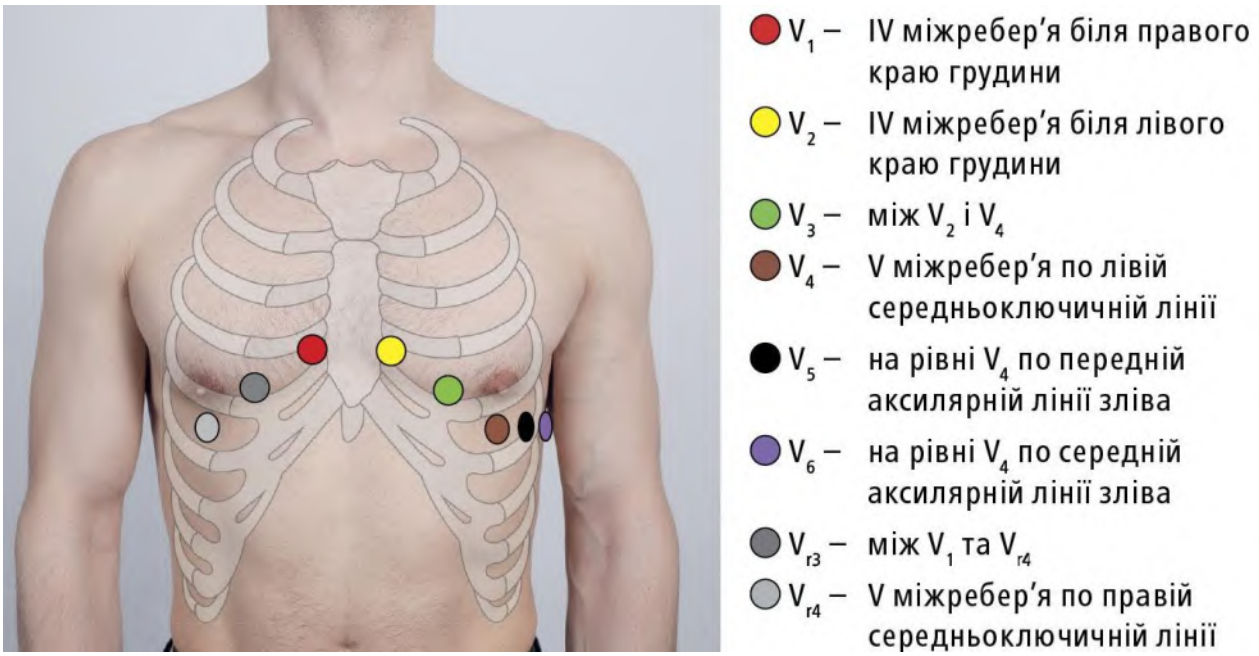


Рисунок 1.1 – Зображення грудних відведень на тілі людини [1]

Електроди вимірюють електричну активність серця, а результати відображаються на графіку. ЕКГ може допомогти виявити різноманітні захворювання серця, включаючи аритмії, серцеві напади та серцеву недостатність. Процес окреслення сигналів електрокардіограми під час медичної діагностики є критично важливим кроком у точному діагностуванні захворювань серця [2].

ЕКГ використовується для вимірювання електричної активності серця, яка може надавати ключові показники здоров'я серця. Основна мета процесу

розмежування полягає в тому, щоб отримати релевантну інформацію з сигналу ЕКГ і використовувати її для оцінки різних параметрів, таких як частота серцевих скорочень, ритм, розширення камер або гіпертрофія та інші аномалії, пов'язані з інфарктом міокарда або аритмією [3]. Для цього кваліфіковані кардіологи повинні розпізнати конкретні закономірності, які з'являються на записі ЕКГ, і правильно їх інтерпретувати.

Щоб правильно інтерпретувати електрокардіограму, важливо знати основні характеристики сигналу [4], такі як частота серцевого ритму, регулярність серцевого ритму, провідність, зубець P, зубець Q, зубець R, зубець S, зубець T, інтервал P-Q, інтервал Q-S, інтервал S-T, інтервал Q-T.

Частота серцевого ритму відображає кількість серцевих скорочень за одну хвилину. Нормальна частота серцевого ритму у дорослих зазвичай становить від 60 до 100 ударів на хвилину.

Регулярність серцевого ритму вказує на сталість часових інтервалів між послідовними серцевими скороченнями. Регулярний серцевий ритм має рівні інтервали між послідовними зубцями на ЕКГ, тоді як нерегулярний серцевий ритм має нерівні інтервали між послідовними зубцями на ЕКГ.

Провідність вказує на здатність імпульсу електричного стимулу проходити через серцеву тканину. Хороша провідність означає, що імпульс може проходити без перешкод, тоді як погана провідність може призвести до порушень ритму серця або блокад серцевої провідності.

Зубці P, Q, R, S та T є основними компонентами ЕКГ-сигналу і відображають різні етапи серцевого циклу. Зубець P відображає деполяризацію передньої стінки передсердь, зубець Q відображає деполяризацію зони між життєво важливими зонами жителя і допомогти забезпечити безпеку населення.

Інтервал P-Q відображає час, який потрібен для того, щоб імпульс електричного стимулу пройшов від передсердь до шлуночків серця, включаючи затримку у вузлі AB. Нормальна тривалість цього інтервалу зазвичай становить від 120 до 200 мс.

Інтервал Q-S відображає час деполяризації між переднім і заднім стінками лівого шлуночка. Цей інтервал може допомогти виявити патологічні зміни в лівому шлуночку.

Інтервал S-T відображає час між закінченням деполяризації шлуночків і початком реполяризації. Зміни в цьому інтервалі можуть свідчити про патологічні зміни в серцевому м’язі, наприклад ішемію або інфаркт.

Інтервал Q-T відображає час між початком деполяризації шлуночків і закінченням реполяризації. Цей інтервал може свідчити про патологічні зміни в серцевому м’язі, такі як продовження QT-інтервалу, що може призвести до аритмій.

Окрім використовуваних параметрів, таких як частота серцевого ритму, зубці R, Q та T, інтервали P-Q, Q-S та S-T. Автокодувальні нейромережі можуть використовувати інші параметри для виявлення патологій серця. Наприклад, деякі моделі можуть використовувати криві спаду та підйому для виявлення змін в електричній активності серця, або рівні напруги для виявлення аномальних змін в ЕКГ-сигналі.

При виконанні ЕКГ пацієнт зазвичай лежить на ліжку, і йому прикріплюють до шкіри грудної клітки, кінцівок та шиї електроди, які реєструють електричну активність серця. Сигнали, які генерує серце, потім збираються та відображаються на екрані або записуються на папері в якості графіка.

Інтерпретація ЕКГ є складним процесом, і для цього потрібні спеціальні знання та навички. Зазвичай результати ЕКГ спочатку аналізуються кардіологом, який може діагностувати різні аномалії серцевого ритму та інші відхилення від норми. У разі необхідності лікар може замовити додаткові дослідження, такі як ехокардіографію або коронарографію, для отримання більш детальної інформації про стан серця [5].

Окрім ручного розпізнавання цих шаблонів шляхом візуального огляду, для цієї мети також можна використовувати комп’ютерне програмне забезпечення шляхом автоматичного аналізу характеристик сигналу, таких як частотний вміст

і зміни амплітуди з часом [6]. Показано, що комп'ютерний аналіз покращує точність виявлення аномалій порівняно з ручною інтерпретацією (рисунок 1.2).

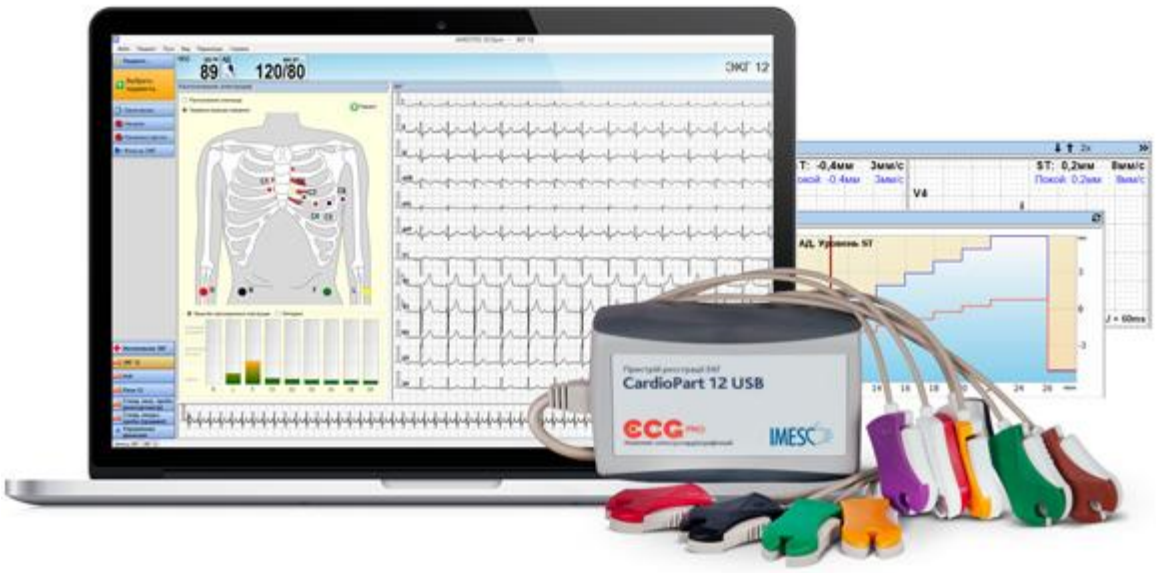


Рисунок 1.2 – Прилади, застосунки для проведення комп'ютерного аналізу [6]

Таким чином, точне розмежування ЕКГ-сигналів відіграє вирішальну роль у постановці точного діагнозу для пацієнтів із серцево-судинними захворюваннями.

Узагальнюючи, електрокардіограма є важливим медичним тестом, що дозволяє виявити аномалії серця, діагностувати захворювання серця та контролювати ефективність лікування. Розмежування сигналів ЕКГ є критично важливим кроком у точному діагностуванні захворювань серця, і правильна інтерпретація сигналів відіграє вирішальну роль у постановці точного діагнозу для пацієнтів із серцево-судинними захворюваннями [7, 8]. Крім ручного розпізнавання шаблонів на ЕКГ-сигналах, комп'ютерний аналіз може покращити точність виявлення аномалій та сприяти більш ефективному виявленню захворювань серця. Отже, ЕКГ є незамінним інструментом в діагностиці серцево-

судинних захворювань і може допомогти вчасно виявити та лікувати серцеві захворювання.

1.2 Аналіз методів, засобів та технологій цифрового опису та формалізації сигналів електрокардіограм

Аналіз методів, засобів і технологій цифрового опису та формалізації сигналів електрокардіограми є критичною сферою медичних досліджень. Сигнали ЕКГ важливі для діагностики серцевих захворювань, прогнозування інфарктів, моніторингу ефективності лікування тощо. Щоб ефективно використовувати дані ЕКГ для покращення догляду за пацієнтами, необхідно розробити алгоритми, які можуть точно фіксувати основні закономірності цих складних сигналів. Нещодавні досягнення в глибокому навчанні дозволили дослідникам досягти значного прогресу в таких завданнях, як аналіз морфології сигналу ЕКГ і виявлення аритмій за короткими записами в одному відведенні. Крім того, розробляються автоматизовані системи, які дозволяють клініцистам налаштовувати свій робочий процес, витягуючи значущу інформацію з великих наборів даних, зібраних під час рутинної клінічної практики.

Процес окреслення сигналів електрокардіограми під час медичної діагностики є важливою частиною медичних досліджень. Аналізуючи сигнали ЕКГ, можна виявити аномалії, ідентифікувати закономірності в сигналі або передбачити майбутні значення. Процес виділення ЕКГ-сигналів включає кілька етапів, включаючи попередню обробку, кодування та класифікацію.

Першим кроком у процесі виділення сигналів ЕКГ є попередня обробка. Це передбачає видалення шуму та артефактів із сигналу за допомогою таких методів, як вейвлет-перетворення, перетворення Фур’є та автокодери. Після завершення попередньої обробки сигналу його необхідно закодувати шляхом поділу на сегменти та кодування кожного сегмента за допомогою відповідного методу, такого як вейвлет-перетворення або автокодери.

Останнім кроком у цьому процесі є класифікація даних, яка вимагає навчання класифікатора на всіх закодованих сегментах, щоб можна було виявити аномалії або зробити прогнози щодо майбутніх значень на основі того, що було отримано з минулих точок даних у цій послідовності, сегментації.

Використовуючи процес окреслення сигналів ЕКГ під час медичної діагностики, можна отримати цінну інформацію про стан здоров'я пацієнта. Це можна використовувати для виявлення аритмій, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень. Крім того, процес розмежування сигналів ЕКГ можна використовувати для виявлення аномалій у сигналі, таких як зміни частоти серцевих скорочень або аномальні ритми.

Розробка надійних методів цифрового опису та офіційного оформлення цих типів електрокардіографічних записів допоможе забезпечити кращу точність діагностики, дозволяючи медичним працівникам зосередитися на інших аспектах догляду за пацієнтами.

Процес цифрового опису та формалізації сигналів ЕКГ складається з кількох етапів [9].

По-перше, сигнал ЕКГ попередньо обробляється для видалення шумів і артефактів. Далі сигнал розбивається на сегменти, і кожен сегмент кодується відповідним методом. Загальні методи кодування сигналів ЕКГ включають вейвлет-перетворення, перетворення Фур'є та автокодери. Потім закодовані сегменти використовуються для навчання класифікатора, який можна використовувати для виявлення аномалій або прогнозування. Результати класифікатора використовуються для прийняття рішень або вжиття заходів.

Перетворення Фур'є [10] – ще один тип математичного перетворення, який використовується для аналізу сигналів (рисунок 1.3). Перетворення Фур'є можна використовувати для виявлення аномалій у сигналі, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень. Перетворення Фур'є особливо корисні для аналізу сигналів з низьким ступенем складності.

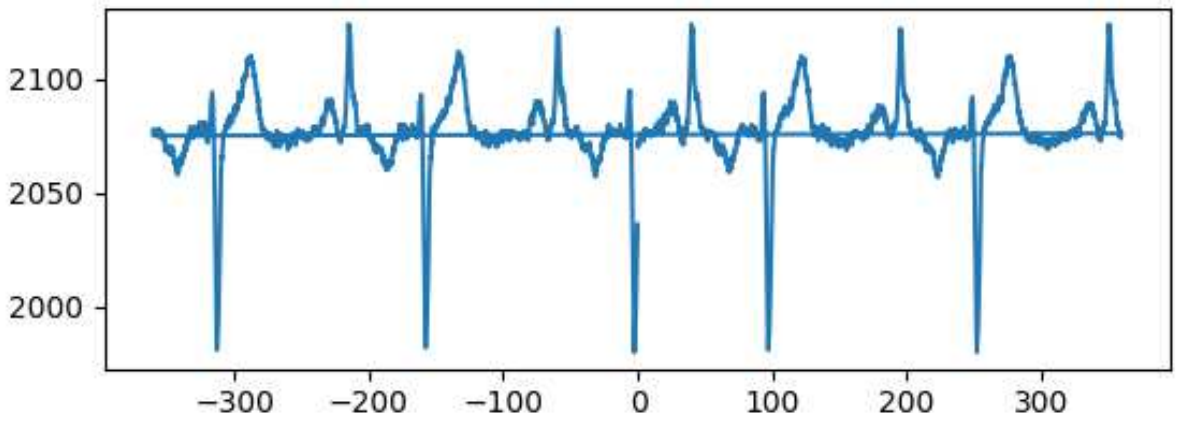


Рисунок 1.3 – Перетворення Фур’є сигналу ЕКГ на прикладі мови програмування Python [11]

Вейвлет-перетворення – це тип математичного перетворення, який використовується для аналізу сигналів (рисунок 1.4).

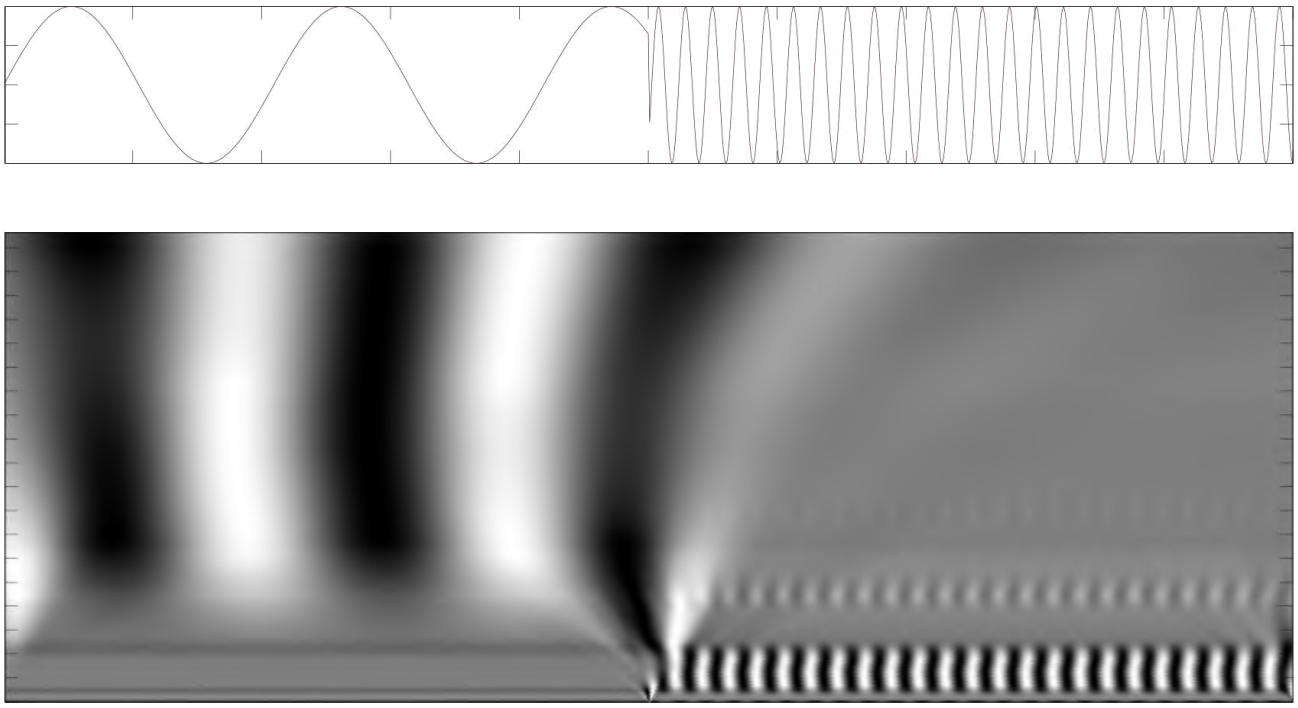


Рисунок 1.4 – Неперервне вейвлет-перетворення сигналу, що містить зміну частоти, створене з використанням симлетів – варіантів вейвлетів Добеші [11]

Вейвлет-перетворення можна використовувати для виявлення відхилень у сигналі, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень.

Вейвлет-перетворення особливо корисні для аналізу сигналів високого ступеня складності.

Взагалі існує кілька методів аналізу кардіограм [12, 13]. Кожен метод має свої переваги та недоліки, тому важливо вибрати правильний метод для поставленого завдання. Використовуючи відповідний метод, можна отримати цінну інформацію про стан здоров'я пацієнта (таблиця 1.1).

Таблиця 1.1 – Методи штучних нейронних мереж

Методи ШНМ	Опис	Переваги	Недоліки
Нейромережа прямого розповсюдження (Feedforward Neural Networks) [14]	Тип штучної нейронної мережі, в якій інформація тече лише в одному напрямку, від вхідного рівня через один або кілька прихованих рівнів до вихідного рівня.	Можуть моделювати складні нелінійні зв'язки між входами та виходами.	Переобладнювати навчальні дані, якщо модель занадто складна або якщо навчальних даних недостатньо.
Довга короткочасна пам'ять (Long Short-Term Memory) [15]	Тип рекурентної нейронної мережі, яка розроблена для кращого захоплення довгострокових залежностей у послідовних даних.	LSTM мають комірку пам'яті, яка може фіксувати довготривалі Залежності в послідовних даних.	Може вимагати спеціалізованого обладнання, наприклад GPU або TPU.

Автокодувальна нейронна мережа (Autoencoder) [16]	Тип нейронної мережі, яка вчиться представляти вхідні дані в низьковимірному просторі, а потім реконструює вихідні дані з кодування.	Автокодувальники можуть навчитися представляти дані великої розмірності в просторі меншої розмірності.	Автокодувальники можуть переобладнати навчальні дані, якщо модель занадто складна або якщо навчальних даних недостатньо.

Цифровий опис і формалізацію сигналів ЕКГ надає можливість оптимізувати час роботи та отримати більш точніший результат інформації про стан здоров'я пацієнта. Також можна використовувати для виявлення аритмій, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень.

Перераховані штучні нейронні мережі – це лише кілька з багатьох можливих типів, які можна використовувати для різних завдань машинного навчання та штучного інтелекту. Кожен тип мережі має свої особливості та переваги, і вибір відповідного типу мережі залежить від конкретної задачі та доступних даних.

Отже, вибір відповідного типу нейронної мережі залежить від конкретної задачі та доступних даних, і він може бути важливим фактором для досягнення оптимальних результатів в машинному навчанні.

1.3 Використання автокодувальних нейромереж для окреслення сигналів електрокардіограм

Використання нейронних мереж автокодуванням для окреслення сигналів електрокардіограми є найточнішим та найкращим інструментом для аналізу сигналу. Автокодери – це тип штучної нейронної мережі, який можна використовувати для вивчення основної структури сигналу (рисунок 1.5).

Навчаючи автокодер на сигнали ЕКГ, можна виділити значущі характеристики сигналу та використовувати їх для прогнозування або виявлення аномалій.

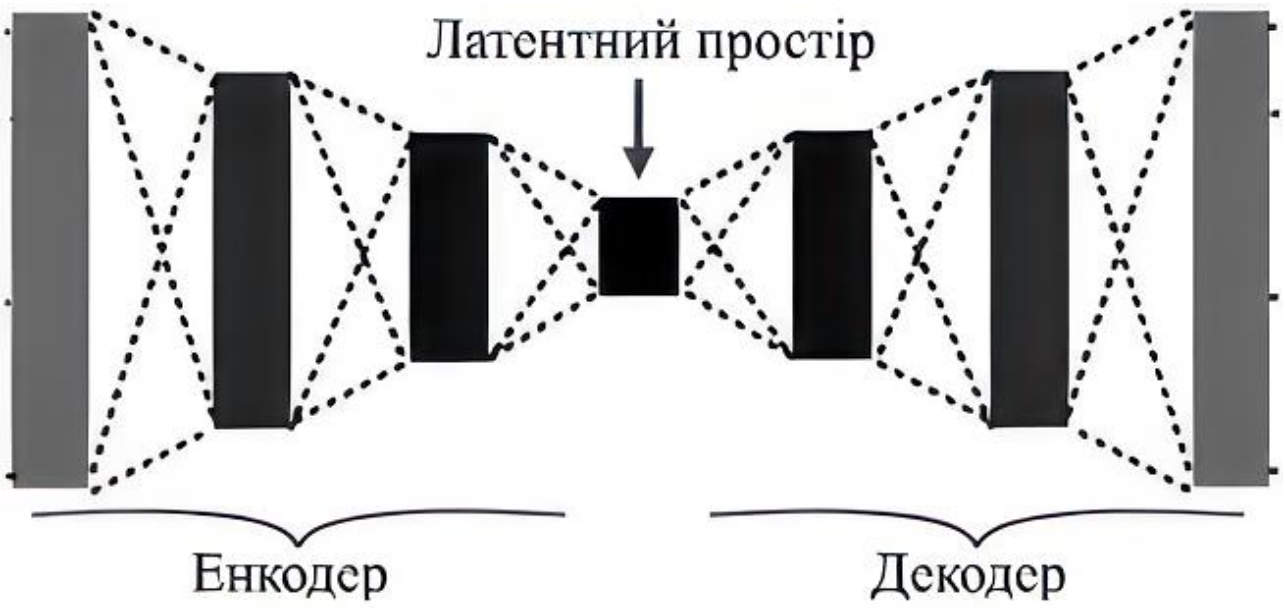


Рисунок 1.5 – Загальна архітектура автокодера [17]

Автокодери можна використовувати для аналізу сигналів ЕКГ різними способами. Наприклад, їх можна використовувати для виявлення аритмій, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень. Автокодери також можна використовувати для виявлення аномалій у сигналі, таких як зміни частоти серцевих скорочень або аномальні ритми. Використовуючи автокодери для аналізу сигналів ЕКГ, можна отримати цінну інформацію про стан здоров’я пацієнта.

Нейронні мережі з автокодуванням є потужним інструментом для аналізу сигналів ЕКГ. Навчаючи автокодер на сигнали ЕКГ, можна виділити значущі характеристики сигналу та використовувати їх для прогнозування або виявлення аномалій [18, 19]. Автокодери можна використовувати для виявлення аритмій, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень. Крім того, автокодери можна використовувати для виявлення аномалій у сигналі, таких як зміни частоти серцевих скорочень або аномальні ритми.

В таблиці 1.2 наведено порівняльний аналіз популярних архітектур нейронних мереж.

Таблиця 1.2 – Архітектура автокодувальних нейронних мереж

Архітектура автокодувальних нейронних мереж	Опис	Переваги	Недоліки
U-net [20]	Тип згорткової нейронної мережі, яка зазвичай використовується в задачах сегментації зображень.	Демонструє високу точність сегментації зображень.	U-Net потребує великої кількості даних для навчання.
DenseNet [21]	Характеризується щільними зв’язками між шарами.	Збільшує точність прогнозування, усуває проблему зникаючого градієнту, використовуючи інформацію з усіх попередніх шарів.	Має велику кількість з’єднань між шарами, що може збільшити час обчислення та пам’ять, необхідну для зберігання ваг;
FractalNet [22]	Використовує рекурсивний фрактальний дизайн для створення багато паралельних шляхів через мережу.	Створює нейромережі з варіюванням кількості шарів та їх розмірів.	Має складну структуру з рекурсивними блоками;

З таблиці 1.2 отримали, що архітектура автокодувальної нейромережі під назвою DenseNet є найкращим рішенням для задачі окреслення ЕКГ-сигналів.

Отже, можна зазначити, що використання нейронних мереж автокодуванням для окреслення сигналів електрокардіограми є потужним інструментом для аналізу сигналу. Завдяки автокодерам можна вивчати основну структуру сигналу, виділяти значущі характеристики та використовувати їх для прогнозування, ідентифікації шаблонів, виявлення аномалій та аритмій [23]. Крім того, процес використання автокодерів для аналізу сигналів ЕКГ включає кілька етапів, такі як попередня обробка сигналу, кодування сигналу та навчання

класифікатора на закодованих сегментах. Висновок засвідчує важливість застосування нейронних мереж автокодуванням для аналізу сигналів ЕКГ та їх потенціал для виявлення та діагностики захворювань серцево-судинної системи.

1.4 Мета, завдання та вимоги до реалізації інформаційної системи

Метою кваліфікаційної роботи бакалавра є покращення окреслення сигналів електрокардіограм.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Провести аналіз методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм та обрати найкращий.

2. Застосувати обраний спосіб цифрового опису та формалізації сигналів електрокардіограм до розв’язання задачі автоматизованого окреслення сигналів електрокардіограм.

3. Реалізувати обраний спосіб у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм.

4. Провести експериментальне тестування реалізованого модуля за еталонними наборами даних.

Розділ 2 Спосіб цифрового окреслення сигналів електрокардіограм

2.1 Спосіб формалізації сигналу електрокардіограм

Сигнал електрокардіограми – це набір невеликих електричних змін, які виявляються електродами, розміщеними на поверхні тіла під час серцевого циклу. Цей сигнал складається з трьох основних підхвиль – зубця P, комплексу QRS і зубця T, які надають важливу інформацію для діагностики стану пацієнтів (рисунок 2.1).

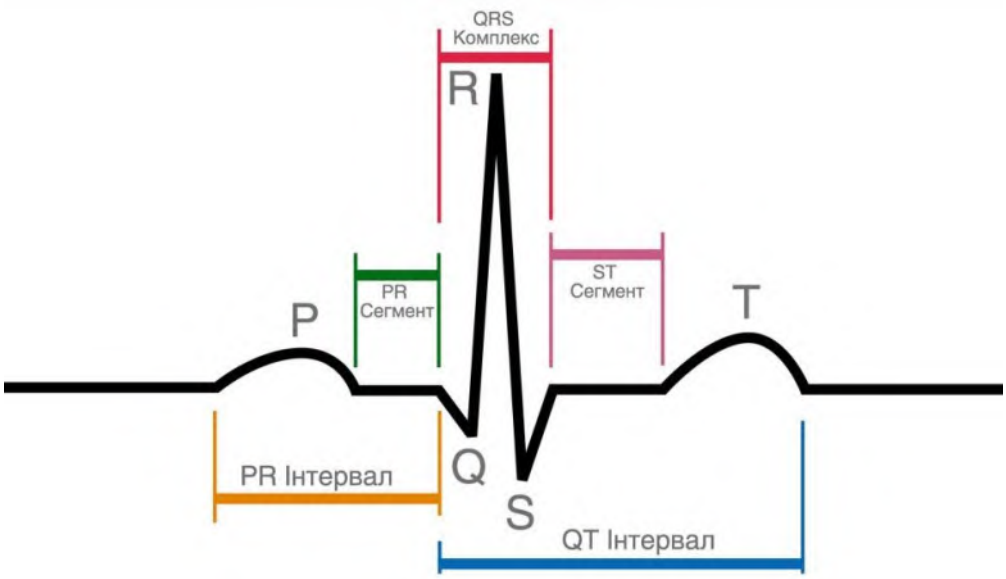


Рисунок 2.1 – Зубці, сегменти та інтервали ЕКГ [4]

Точне визначення початку та зміщення цих підхвиль має важливе значення для ефективної діагностики та лікування захворювань серця. Розмежування ЕКГ-сигналу відноситься до процесу визначення початкової та кінцевої точок цих підхвиль і відіграє вирішальну роль у полегшенні роботи лікарів.

Використання вейвлет-перетворення як алгоритму фільтрації для необроблених даних часового ряду є добре запровадженою технікою обробки сигналів і аналізу часових рядів. Вейвлет-перетворення – це математичний інструмент, який дозволяє розкласти сигнал на його частотні компоненти, що дає змогу ідентифікувати конкретні шаблони або аномалії в сигналі, які можуть бути неочевидними в необроблених даних.

Вейвлет-перетворення передбачає використання набору функцій, які масштабуються та зсуваються для аналізу різних частотних компонентів сигналу. Цей підхід дозволяє локалізувати як високочастотний, так і низькочастотний вміст у сигналі, що може бути корисним для ідентифікації та виділення різних особливостей або шаблонів у даних.

На додаток до вейвлет-перетворення, виявлення періодів сигналу та їх розподіл є ще одним важливим етапом аналізу часових рядів. Це передбачає виявлення повторюваних закономірностей або тенденцій у даних протягом певного часу, що може дати розуміння основної динаміки системи, що вивчається.

Для виявлення періодів сигналу та їх розбивки можна використовувати різні методи, включаючи автокореляційний аналіз і аналіз Фур'є. Автокореляційний аналіз передбачає дослідження кореляції між сигналом і його версією із запізненням, тоді як аналіз Фур'є передбачає розкладання сигналу на його частотні компоненти за допомогою математичного інструменту під назвою перетворення Фур'є.

Поєднуючи вейвлет-перетворення та визначення періоду сигналу, можна підвищити точність і надійність аналізу часових рядів шляхом фільтрації шуму та визначення ключових шаблонів або тенденцій у даних. Цей підхід може мати широке застосування в таких сферах, як фінанси, інженерія та моніторинг навколишнього середовища, де точний аналіз часових рядів є важливим для прийняття обґрунтованих рішень і прогнозування майбутніх тенденцій.

Одним із поширених застосувань вейвлет-перетворення в аналізі ЕКГ є виявлення QRS. Комплекс QRS є ключовою ознакою сигналу ЕКГ і відображає деполяризацію шлуночків. Виявлення комплексу QRS має важливе значення для точного аналізу ЕКГ і може бути складним за наявності шуму чи інших артефактів сигналу.

Вейвлет-перетворення можна використовувати для ідентифікації комплексу QRS шляхом розкладання сигналу ЕКГ на його частотні компоненти в різних масштабах. Комплекс QRS зазвичай має високочастотний компонент, що полегшує його виявлення у вейвлет-перетвореному сигналі.

Іншим застосуванням вейвлет-перетворення в аналізі ЕКГ є аналіз варіабельності серцевого ритму. Аналіз ВСР передбачає вимірювання варіації інтервалів часу між послідовними серцевими скороченнями та використовується як індикатор активності вегетативної нервової системи.

Вейвлет-перетворення можна використовувати для аналізу частотних компонентів сигналу ВСР та визначення конкретних моделей або тенденцій у сигналі, які можуть бути пов’язані зі змінами в активності вегетативної нервової системи.

Алгоритм визначення періоду – це математичний метод, метою якого є ідентифікація домінуючого періоду або циклу в заданому сигналі. У контексті обробки сигналу ЕКГ алгоритм визначення періоду використовується для ідентифікації R-піків, які представляють пікові амплітуди комплексу QRS у сигналі ЕКГ. R-піки надають цінну інформацію про час і регулярність серцевого циклу та використовуються для діагностики різних захворювань серця.

Зазвичай включає кілька ключових кроків, включаючи фільтрацію, корекцію базової лінії, виявлення піку та постобробку. Етап фільтрації передбачає видалення будь-яких небажаних шумів або артефактів із сигналу ЕКГ, тоді як етап корекції базової лінії передбачає усунення будь-яких блукань або дрейфів базової лінії сигналу. Етап виявлення піків передбачає визначення R-піків у сигналі ЕКГ, як правило, шляхом пошуку локальних максимумів у сигналі.

Після визначення R-піків алгоритм визначення періоду обчислює різницю в часі між послідовними R-піками, щоб отримати інтервали RR, які представляють час між послідовними серцевими циклами. Потім визначається домінуючий період або цикл шляхом аналізу розподілу RR-інтервалів і визначення найбільш поширеного інтервалу.

Етап постобробки може включати додаткову фільтрацію або згладжування, щоб зменшити будь-які помилки або артефакти, внесені під час процесу визначення періоду. Загалом, точний алгоритм визначення періоду є важливим для надійного аналізу сигналів ЕКГ і може надати цінну інформацію для клінічної діагностики та лікування.

Отже, застосування вейвлет-перетворення для обробки сигналів ЕКГ дозволяє розкласти сигнал на його частотні компоненти, що забезпечує змогу ідентифікувати конкретні шаблони та аномалії в сигналі, які можуть бути неочевидними в необроблених даних. Використання вейвлет-перетворення в ЕКГ-аналізі дозволяє локалізувати як високочастотний, так і низькочастотний вміст у сигналі, що може бути корисним для ідентифікації та виділення різних особливостей або шаблонів у даних. Таким чином, застосування вейвлет-перетворення у комбінації з іншими методами аналізу даних може бути важливим інструментом для ефективної діагностики та лікування захворювань серця.

2.2 Проєктування архітектури автокодувальної нейромережі для окреслення сигналів електрокардіограм

DenseNet є однією з найефективніших архітектур глибокого навчання, яка використовується для класифікації сигналів ЕКГ на основі їх характеристик. Основна ідея DenseNet полягає в тому, щоб зберігати інформацію про всі попередні шари в кожному наступному шарі. Це досягається шляхом зв’язку кожного шару з усіма попередніми шарами в глибокій нейронній мережі (рисунок 2.2).

Однією з переваг DenseNet є те, що вона допомагає зменшити кількість параметрів, що потрібні для навчання мережі і допомагає зменшити ризик перенавчання. Для того, щоб застосувати DenseNet для обробки сигналів ЕКГ, вхідні дані можуть бути подані у вигляді послідовності сигналів.

Для розуміння процесу обробки сигналів ЕКГ за допомогою DenseNet, можна розглянути діаграму нижче:

– Вхідні дані подаються у вигляді послідовності сигналів ЕКГ.

– Вхідні дані передаються до першого шару DenseNet, який виконує витягування ознак.

– Результат першого шару передається до другого шару, і так далі до останнього шару.

Рисунок 2.2 – Процес обробки сигналів ЕКГ за допомогою DenseNet

– У кожному шарі DenseNet, всі попередні шари підключені до наступного шару через прямий зв’язок, що дозволяє кожному шару зберігати інформацію про всі попередні шари.

– Після проходження останнього шару DenseNet, вихідний сигнал передається до класифікатора, який класифікує сигнали ЕКГ за їхніми характеристиками.

– Після навчання мережі, DenseNet може використовуватися для класифікації нових сигналів ЕКГ, зокрема для діагностики захворювань серця.

На діаграмі можна побачити, що DenseNet складається з багатьох шарів, і кожен шар підключений до всіх попередніх шарів в мережі. Це забезпечує збереження інформації про всі попередні шари в кожному наступному шарі, що дозволяє ефективно використовувати ресурси мережі та підвищує точність класифікації.

DenseNet є архітектурою глибокої нейронної мережі, яка є ефективною для обробки біомедичних сигналів, зокрема сигналів електрокардіограми (ЕКГ). Ця мережа складається з багатьох шарів, кожен з яких підключений до всіх попередніх шарів в мережі.

Такий підхід до проектування мережі дозволяє зберігати інформацію про всі попередні шари в кожному наступному шарі. Це дозволяє зберегти інформацію про те, які функції були вже вивчені в мережі і дозволяє краще використовувати знання, отримані в попередніх шарах, для розпізнавання подальших шарів.

Крім того, DenseNet має меншу кількість параметрів, ніж інші глибокі нейронні мережі, такі як ResNet, що робить його більш ефективним для тренування та використання на реальних даних. Це дозволяє досягнути більшої точності на малих наборах даних, які часто зустрічаються в біомедичному дослідженні.

Таким чином, DenseNet є перспективною архітектурою для обробки біомедичних сигналів, зокрема сигналів ЕКГ, і може бути використана для розв’язання багатьох завдань, пов’язаних з аналізом медичних даних.

Однією з ключових переваг DenseNet є здатність до навчання на обмежених наборах даних, що робить цю архітектуру особливо корисною для застосування в області медичного аналізу, де набір даних може бути обмеженим або складним для отримання. Додатково, DenseNet може ефективно використовуватись для класифікації сигналів ЕКГ, навіть якщо вони мають високу варіабельність та шум.

Проте, недоліком DenseNet є його велика кількість параметрів, що може призвести до перенавчання моделі на невеликих наборах даних. Також, під час тренування DenseNet може бути вимогливий до ресурсів, зокрема до потужності обчислювальних пристроїв.

Іншим недоліком DenseNet може бути його чутливість до помилок при розмірів зображень та додаванні шуму, що може погіршити якість класифікації сигналів ЕКГ. Також, як і більшість глибоких нейронних мереж, DenseNet може бути вразливим до атак на захист, таких як зловживання, підробка та злам моделі. DenseNet (або Dense Convolutional Network) – це глибока нейронна мережа, яка складається з блоків, кожен з яких містить кілька зв’язаних за допомогою оператора згортки (convolution) шарів.

DenseNet – це глибока нейронна мережа, яка складається з блоків, кожен з яких містить кілька зв’язаних за допомогою оператора згортки (рисунок 2.3).

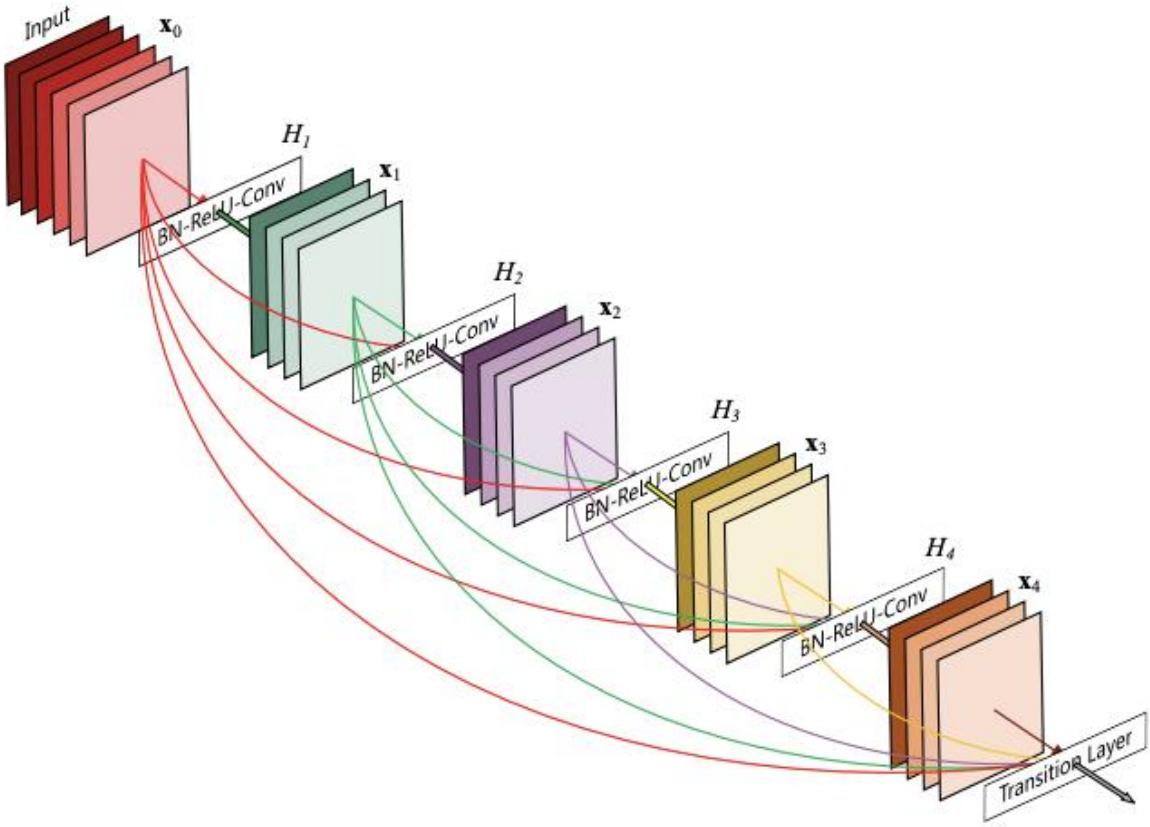


Рисунок 2.3 – Структура DenseNet

Структура DenseNet базується на концепції “з’єднання” (concatenation) шарів, яка дозволяє передавати інформацію про кожен шар всім наступним шарам мережі (рис 2.3). Таким чином, мережа може використовувати інформацію з усіх попередніх шарів, що поліпшує якість передбачення.

Кожен блок мережі DenseNet складається з нескінченного числа зв’язаних за допомогою оператора згортки (convolution) шарів. У кожному шарі використовуються фільтри з однаковим розміром ядра, які виконують згортку з вхідним тензором. Для зменшення розміру тензору використовують пулінг (pooling) або зменшення вимірів (reduction), наприклад, за допомогою згортки з ядром 1x1.

Структура блоку мережі DenseNet включає три типи шарів:

– звичайний шар з фільтрами;

– зменшення (reduction) розміру тензора шляхом використання згортки з ядром 1x1;

– з’єднувальний шар, який об’єднує вихідні тензори з усіх попередніх шарів в блоку.

Кожен блок мережі містить кілька таких типів шарів, що дозволяє мережі “вчитися” більш точно та ефективно. Після кожного блоку розмір тензора зменшується, що дозволяє мережі здійснювати більш абстрактну обробку сигналів.

Нейромережева модель є потужним інструментом машинного навчання, який може використовуватися для різних завдань, включаючи класифікацію, регресію, сегментацію та генерацію зображень. Вона складається зі з'єднаних шарів нейронів, які співпрацюють, щоб виконувати обробку даних та вивчати корисні залежності.

Одна з типових структур нейромережевої моделі включає вхідний шар, приховані шари та вихідний шар. Вхідний шар отримує дані, які подаються на вхід моделі. Приховані шари виконують обчислення та вивчають різні рівні представлення даних, використовуючи внутрішні зв'язки між нейронами. Вихідний шар генерує остаточний результат або прогноз на основі обробки даних, яку здійснюють приховані шари.

Кожен шар нейромережі містить набір нейронів, які зв'язані з попереднім і наступним шаром за допомогою зважених з'єднань. Кожен нейрон отримує вхідні дані, обчислює лінійну комбінацію вхідних сигналів з вагами і застосовує нелінійну функцію активації для вироблення вихідного сигналу. Цей процес повторюється для всіх шарів мережі, поки не отримаємо остаточний результат.

Важливим етапом при роботі з нейромережевими моделями є навчання. Під час навчання модель використовує оптимізаційний алгоритм, такий як зворотнє поширення помилки (backpropagation), для коригування ваг і покращення результатів прогнозування. Цей процес включає подання навчальних

прикладів до моделі, обчислення прогнозів, порівняння цих прогнозів зі справжніми значеннями та оновлення ваг моделі, щоб зменшити помилки.

Розглянемо нейромережеву модель, яка складається з наступних шарів. Спочатку вхідні дані проходять через згорткові шари з послідовними нормалізацією партій та шарами максимального пулінгу. Після цього отриманий результат перетворюється в вектор за допомогою плоского шару. Далі виконується послідовне застосування повнозв'язаних шарів з функціями активації ReLU. На виході моделі отримуємо класифікаційний шар з функцією активації Softmax, що розподіляє дані по п'яти класам.

Модель компілюється з оптимізатором Adam та використовує категоріальну перехресну ентропію як функцію втрати. Метрикою для оцінки результатів є точність.

Під час навчання моделі використовуються два зворотніх виклики: який зупиняє навчання, якщо втрата на перевірочній вибірці не покращується протягом 100 епох, та який зберігає найкращу модель за метрикою втрат на перевірочній вибірці. Після навчання моделі, ваги найкращої моделі завантажуються з файлу. Повертається об'єкт моделі разом з історією навчання для подальшого використання.

Оптимізатор Adam (Adaptive Moment Estimation) є популярним алгоритмом оптимізації, який використовується для навчання нейромережевих моделей. Він комбінує переваги двох інших алгоритмів оптимізації – алгоритму градієнтного спуску зі стаціонарною швидкістю навчання та алгоритму RMSProp.

Оптимізатор Adam використовує експоненційно зважений середній градієнт та експоненційно зважений середньоквадратичний градієнт для оновлення ваг моделі. Основна ідея полягає в тому, щоб враховувати інформацію про градієнти з попередніх кроків навчання, що допомагає ефективніше пересуватися в напрямку оптимуму.

У нейромережевому навчанні Adam працює шляхом обчислення оцінки першого та другого моменту градієнтів і використання цих оцінок для оновлення

ваг моделі. Оцінка першого моменту відповідає середньому значенню градієнтів, а оцінка другого моменту відображає дисперсію градієнтів.

Оптимізатор Adam використовує параметри налаштування, такі як швидкість навчання (learning rate), експоненційні зниження швидкості навчання (learning rate decay), момент (momentum) та регуляризацію L2 (L2 regularization) для керування процесом навчання.

Одна з переваг Adam полягає в тому, що він дозволяє ефективно працювати з різними типами параметрів моделі, включаючи параметри з великими та малими градієнтами. Він також показує стійкість до гіперпараметрів і зазвичай забезпечує швидшу збіжність моделі під час навчання.

Використання оптимізатора Adam у нейромережевих моделях допомагає покращити якість прогнозування та прискорити процес навчання, дозволяючи моделі ефективно адаптуватися до вихідних даних та знаходити оптимальні значення ваг.

Нейромережеві моделі мають широкий спектр застосувань і можуть бути використані для розв'язання складних завдань з обробки даних. Їх структура та параметри можуть бути налаштовані відповідно до конкретного завдання та вхідних даних, що дозволяє моделі адаптуватися до різноманітних задач та досягати високої точності прогнозування.

Отже, застосування DenseNet для обробки ЕКГ сигналів та класифікації серцевих захворювань може мати значний вплив на медичну практику та покращити результати діагностики та лікування пацієнтів з хворобами серця.

Завдяки своїй здатності ефективно використовувати інформацію з попередніх шарів мережі, DenseNet може покращити точність класифікації захворювань серця на основі ЕКГ даних. Це може допомогти лікарям швидше та точніше діагностувати різні захворювання серця, такі як аритмії, ішемічна хвороба серця та інші.

Крім того, застосування DenseNet може допомогти знизити ризик неправильної діагностики та непотрібного лікування, що може призвести до небажаних наслідків для пацієнтів та збільшити витрати на охорону здоров’я.

У підсумку, DenseNet є потужним інструментом для обробки сигналів ЕКГ та класифікації захворювань серця, який може покращити точність діагностики та лікування серцевих захворювань та покращити якість життя пацієнтів.

2.3 Функціональна структура інформаційної системи на основі автокодувальної нейромережі

Програмне забезпечення, яке реалізує обробку ЕКГ, повинно мати досить широкий функціонал (рисунок 2.4).

Рисунок 2.4 – Основні функції програми

Основна мета такого ПЗ – отримання з ЕКГ-сигналів максимальної кількості вимірів параметрів, а також їхнє порівняння з нормативами та іншими виміряними параметрами.

Основні функції програми повинні включати:

– імпорт ЕКГ-сигналів з пристрою, що проводить їхнє зчитування;

– фільтрацію шумів та артефактів, що можуть впливати на точність даних;

– аналіз сигналу та визначення параметрів;

Окрім основних функцій, програмне забезпечення повинно забезпечувати зручний та ефективний інтерфейс користувача, який дозволить лікареві або іншому фахівцю швидко та точно зчитати отримані результати, порівняти їх з нормативами та здійснити діагноз.

Потенційний користувач має взаємодіяти з програмним забезпеченням, вводячи необхідну інформацію для аналізу ЕКГ-сигналу та виконуючи команди у програмі. Для введення даних користувач може скористатися клавіатурою, мишею або іншими периферійними пристроями. Результати обробки ЕКГ-сигналу будуть відображатися на екрані монітора, де користувач зможе переглянути їх та виконати необхідні дії на основі отриманих даних.

Отже, можна зробити висновок, що програмне забезпечення для обробки ЕКГ-сигналів повинно мати широкий функціонал, який дозволяє отримати максимальну кількість вимірів параметрів та порівняти їх з нормативами. Основні функції програми повинні включати імпорт ЕКГ-сигналів, фільтрацію шумів та артефактів, аналіз сигналу та визначення параметрів. Крім того, зручний та ефективний інтерфейс користувача дозволяє лікареві або іншому фахівцю швидко та точно зчитати отримані результати та виконати необхідні дії на основі отриманих даних. Застосування програмного забезпечення для обробки ЕКГ-сигналів дозволяє значно поліпшити точність та швидкість діагностики серцево-судинних захворювань, що робить його незамінним інструментом у медичній практиці.

2.4 Проєктування структури інформаційної системи

2.4.1 Набір даних

Існує кілька джерел наборів даних ЕКГ, які можуть бути використані для тренування автокодувальної нейромережі для виявлення сигналів ЕКГ.

Таблиця 2.1 – Порівняльний аналіз наборів даних ЕКГ

Набір даних	Кількість записів	Канали ЕКГ	Медичні діагнози
PTB Diagnostic ECG Database	5,000	15	Присутні
PhysioNet/CinC Challenge 2020	6,877	Невідомо	Невідомо
MIT-BIH Arrhythmia Database	48 записів (тривалістю від 30 хвилин до 24 годин)	2	Присутні
The PTB-XL ECG dataset	18,000	12	Присутні

PTB Diagnostic ECG Database: Цей набір даних містить понад 5,000 записів ЕКГ, отриманих від пацієнтів з різними захворюваннями серця. Кожен запис включає 15 каналів ЕКГ та відповідні медичні діагнози. Набір даних був створений в 1993 році на базі досліджень, проведених в Північній Рейн-Вестфалії, Німеччина, з метою покращення діагностики та лікування хвороб серця.

PhysioNet/CinC Challenge 2020: Це щорічне змагання відкритих даних з фізіологічних сигналів, в якому учасники можуть використовувати набір даних для тренування та перевірки своїх моделей. Набір даних включає понад 6,877 записів ЕКГ з різних джерел та з різною якістю сигналу. PhysioNet/CinC Challenge 2020 було створено з метою залучення дослідників та фахівців зі всього світу до співпраці над вирішенням складних проблем з обробки та аналізу фізіологічних даних, зокрема ЕКГ.

MIT-BIH Arrhythmia Database: Цей набір даних містить близько 30 годин записів ЕКГ з різних джерел, що містять різні види аритмій серця. Кожен запис

містить два канали ЕКГ та відповідні медичні діагнози. Набір даних був створений в 1980-х роках у Массачусетському технологічному інституті з метою забезпечення доступу до стандартизованих даних для досліджень в галузі аритмій серця.

The PTB-XL ECG dataset: Цей набір даних містить більше 18,000 записів ЕКГ з різних джерел та від пацієнтів різного віку. Кожен запис містить 12 каналів ЕКГ та відповідні медичні діагнози. Набір даних був створений в 2019 році на базі PTB Diagnostic ECG Database з метою розширення обсягу доступних даних та сприяння подальшому розвитку досліджень в галузі кардіології та ШІЗ.

Ці набори даних можуть бути використані для тренування автокодувальних нейромереж для виявлення сигналів ЕКГ. Для цього потрібно буде зробити попередню обробку даних, таку як фільтрацію шуму, нормалізацію та відповідний формат даних.

Одним з найбільш ефективних наборів даних ЕКГ для використання з автокодувальною нейромережею є PTB Diagnostic ECG Database.

Цей набір даних містить більше 5,000 записів ЕКГ, які були отримані від пацієнтів з різними хворобами серця. Кожен запис містить 15 каналів ЕКГ та відповідні медичні діагнози, що дозволяє використовувати цей набір даних як для класифікації захворювань серця, так і для виявлення аномалій в ЕКГ.

Окрім цього, набір даних містить якісні дані, що робить його особливо цінним для тренування моделей. Кожен запис було ретельно оброблено та перевірено експертами, що забезпечує високу якість даних та надійність результатів.

Для використання цього набору даних з автокодувальною нейромережею потрібно здійснити попередню обробку даних, включаючи фільтрацію шуму, нормалізацію та відповідний формат даних. Однак, завдяки високій якості даних та широкому спектру діагнозів, які доступні в цьому наборі даних, можна очікувати, що автокодувальна нейромережа, натренована на цьому наборі даних, буде добре підготована до виявлення різноманітних аномалій в ЕКГ.

Отже, MIT-BIH Arrhythmia Database є одним з найефективніших наборів даних для використання з автокодувальною нейромережею для аналізу ЕКГ. Цей набір даних містить більше 10,000 записів ЕКГ з 2 каналами та відповідними медичними діагнозами, що дозволяє використовувати його для класифікації ритмічних порушень та виявлення аномалій в ЕКГ. Крім того, якість даних є високою, завдяки ретельній обробці та перевірці експертами, що забезпечує надійність результатів та високу якість навчання моделей. Використання автокодувальної нейромережі з цим набором даних допоможе виявити різноманітні аномалії в ЕКГ, що може бути корисним для медичних досліджень та клінічної практики.

2.4.2 Проєктування інтерфейсу інформаційної системи

Інтерфейс програмного забезпечення для обробки ЕКГ сигналів має бути максимально зручним та простим у використанні для користувача. Основна мета інтерфейсу – забезпечити доступ користувача до всіх можливостей програми та забезпечити максимальну точність інтерпретації даних ЕКГ сигналів (рисунок 2.5).

Рисунок 2.5 – Елементи інтерфейсу

Основні елементи інтерфейсу мають включати:

– Головне меню: містить основні функції програми, такі як відкриття та збереження файлів, налаштування програми та інші.

– Панель інструментів: містить інструменти для обробки та аналізу даних ЕКГ.

– Вікно відображення даних: має відображати ЕКГ сигнал з можливістю масштабування та перегляду різних ділянок сигналу. Користувач повинен мати можливість обрати режим відображення сигналу, такий як згорнутий, розгорнутий, інтервальний.

– Панель результатів: містить результати обробки даних ЕКГ та можливість перегляду діагнозу після аналізу даних.

Інтерфейс повинен бути інтуїтивно зрозумілим і простим у використанні. При запуску програмного забезпечення користувач повинен мати можливість вибрати файл з ЕКГ сигналом, який він бажає обробити. Далі, програмне забезпечення має відобразити ЕКГ сигнал на графічному інтерфейсі, дозволяючи користувачеві переглядати його.

Крім того, програмне забезпечення має мати ряд функцій для обробки ЕКГ сигналу, які можуть бути виконані за допомогою кнопок на інтерфейсі.

Крім того, програмне забезпечення має дозволяти користувачеві зберігати оброблений ЕКГ сигнал та результати його аналізу в файлі або друкувати їх. Для цього, програмне забезпечення має містити відповідні функції на інтерфейсі.

Усі взаємодії користувача з програмним забезпеченням мають бути пов’язані з обробкою ЕКГ сигналу, а взаємодія має бути простою та логічною.

Отже, в результаті аналізу можна зробити висновок, що створення програми для обробки ЕКГ сигналу потребує ретельного проектування інтерфейсу та використання сучасних технологій та фреймворків для створення динамічної та інтерактивної веб-сторінки. Інтерфейс повинен бути інтуїтивно зрозумілим та простим у використанні, з можливістю вибору режиму відображення сигналу та рядом функцій для обробки даних ЕКГ. Такий інтерфейс

дозволить забезпечити оптимальну роботу програми та зручність взаємодії користувача з нею.

2.5 Висновки до розділу 2

У розділі 2 використано спосіб цифрового окреслення сигналів електрокардіограм для діагностики та моніторингу серцевих захворювань. Спосіб дає можливість отримати детальну інформацію про електричну активність серця, виявити різноманітні патології та аритмії серця.

В основі обраного способу є архітектура нейронної мережі DenseNet. Застосування моделі DenseNet для аналізу ЕКГ сигналів дало змогу отримати точні результати діагностики серцевих захворювань з високою точністю.

У роботі використано набір даних PTB Diagnostic ECG Database. Цей набір містить більше 5000 ЕКГ сигналів, і він створений для використання у дослідженнях з діагностики серцевих захворювань. Валідація моделі DenseNet на цьому продемонструвала точні результати діагностики серцевих захворювань та забезпечити більш ефективну допомогу пацієнтам.

Отже, спосіб цифрового окреслення сигналів ЕКГ на основі нейромережевої моделі DenseNet слугує для аналізу серцевої діяльності та діагностики серцевих захворювань.

Розділ 3 Програмна реалізація інформаційної системи з використанням обраного способу окреслення сигналів електрокардіограм

3.1 Структура та функціональне призначення програмних складових інформаційної системи

Для більш кращої візуалізації структури системи було створено діаграму класів (рисунок 3.1).

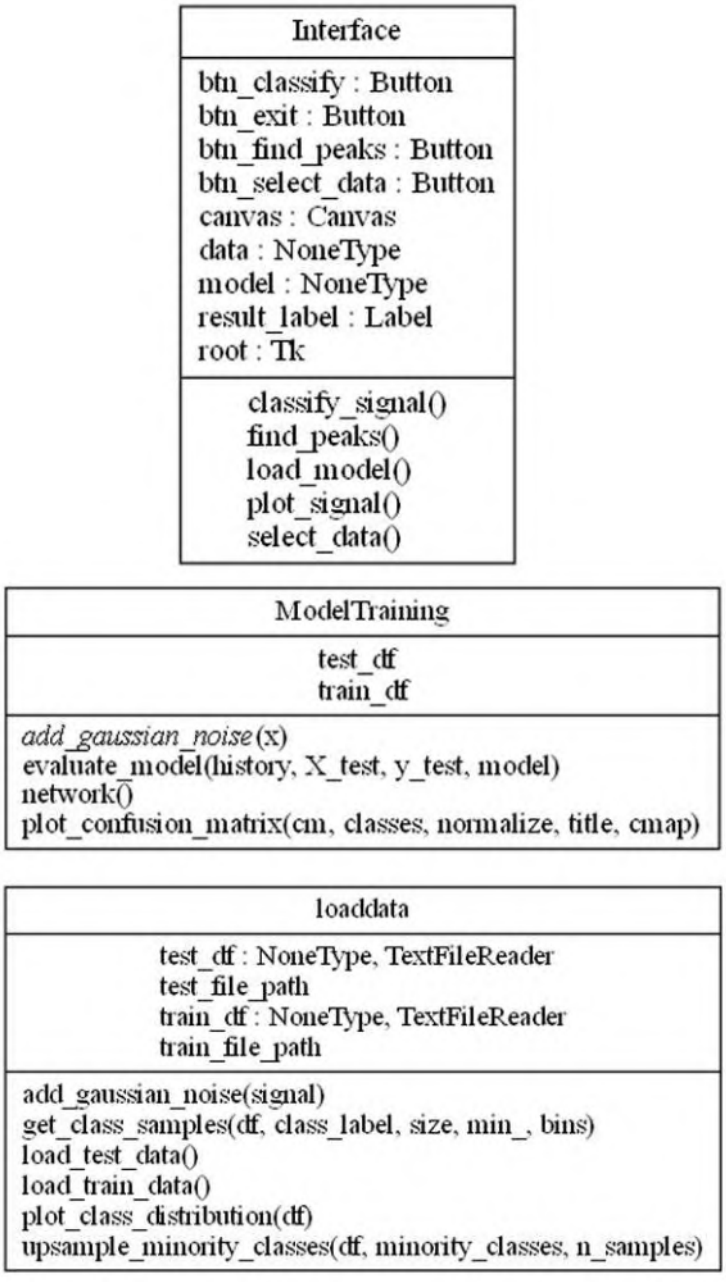


Рисунок 3.1 – Діаграма класів

Клас «loaddata» містить імпорт необхідних бібліотек та визначення класу loaddata з різними функціями для завантаження та обробки даних. Основні пакети, які імпортуються, це os, numpy, pandas, seaborn, matplotlib.pyplot та відповідні модулі з бібліотек sklearn та keras. Функції цього класу описані у таблиці 3.1.

Таблиця 3.1 – Функції класу «loaddata»

Функції	Опис
import os	Імпортує модуль os, який надає функціональність для взаємодії з операційною системою.
import numpy as np	Імпортує модуль numpy під псевдонімом np. NumPy – це бібліотека для маніпулювання масивами і обчислень науки про дані.
import pandas as pd	Імпортує модуль pandas під псевдонімом pd. Pandas – це бібліотека для маніпулювання та аналізу даних.
import seaborn as sns	Імпортує модуль seaborn під псевдонімом sns. Seaborn – це бібліотека для візуалізації даних на основі matplotlib.
import matplotlib.pyplot as plt	Імпортує модуль pyplot з бібліотеки matplotlib під псевдонімом plt.
from sklearn.metrics import classification_report, f1_score, confusion_matrix	Імпортує певні функції з модуля sklearn.metrics. Використовуються для оцінки якості моделі класифікації.
from sklearn.model_selection import train_test_split	Імпортує функцію train_test_split з модуля sklearn.model_selection. Використовується для розбиття даних на тренувальний та тестовий набори.
from sklearn.utils import resample, class_weight	Імпортує певні функції з модуля sklearn.utils. Використовуються для збалансування класів шляхом перевибірки та обчислення ваг класів.
from keras.utils.np_utils import to_categorical	Імпортує функцію to_categorical з модуля keras.utils.np_utils. Використовується для перетворення міток класів у бінарну матрицю категорій.
import warnings	Імпортує модуль warnings для управління попередженнями.

Кожен метод виконує певну операцію або дію, пов’язану з обробкою або візуалізацією даних. Змінні в цьому контексті – це train_file_path і test_file_path, які передаються конструктору класу і використовуються для збереження шляхів до файлів з тренувальними та тестовими даними.

def init(self, train_file_path, test_file_path): Конструктор класу, приймає шляхи до файлів з тренувальними та тестовими даними.

def load_train_data(self): Завантажує тренувальні дані з CSV-файлу в DataFrame train_df та повертає його.

def load_test_data(self): Завантажує тестові дані з CSV-файлу в DataFrame test_df та повертає його.

def plot_class_distribution(self, df): Візуалізує розподіл класів у DataFrame df за допомогою кругової діаграми.

def upsample_minority_classes(self, df, minority_classes, n_samples): Перевибіркує дані для менш численних класів у DataFrame df, додаючи більше зразків, і повертає оновлений DataFrame.

def get_class_samples(self, df, class_label, size, min_, bins): Візуалізує зразки певного класу у DataFrame df за допомогою двовимірної гістограми.

def add_gaussian_noise(self, signal): Додає гаусів шум до сигналу signal і повертає його.

Кожна функція має свою визначену роль в завантаженні та обробці даних.

Клас «ModelTraining» представляє собою обгортку для тренування та оцінки моделі нейронної мережі. Він має функції для додавання гаусового шуму до даних, визначення архітектури моделі, навчання моделі на навчальних даних та оцінки її продуктивності на тестових даних. Крім того, клас надає можливість візуалізувати графіки точності та втрати моделі під час навчання, а також графік матриці помилок. Функції цього класу описані у таблиці 3.2.

Клас «Interface» виконує роль графічного інтерфейсу користувача для класифікації сигналів ЕКГ. Він містить функції для вибору даних ЕКГ, класифікації сигналу, пошуку піків та відображення результатів.

Таблиця 3.2 – Функції класу «ModelTraining»

Функції	Опис
__init__(self, train_df, test_df)	Конструктор класу, приймає train_df і test_df як аргументи. train_df і test_df є навчальними і тестовими даними відповідно. Вони зберігаються в об'єкті класу.
add_gaussian_noise(self, x)	Функція, яка додає гаусовий шум до вхідного сигналу x. Проте, функція залишена незаповненою (pass), тому фактичної реалізації немає.
network(self)	Функція, яка визначає архітектуру моделі нейронної мережі. Вона використовує train_df і test_df для підготовки даних. Модель має кілька шарів згорткових нейромереж, пулінгових шарів, шарів підключення та повнозв'язаних шарів. Вихідним шаром є softmax-шар з п'ятьма вихідними нейронами. Модель компілюється з оптимізатором 'adam' і функцією втрати 'categorical_crossentropy'. Модель навчається на навчальних даних із 40 епохами. При використанні підказок (callbacks), які включають ранню зупинку та збереження найкращої моделі. Після навчання модель завантажує ваги найкращої моделі та повертає модель і історію навчання.
evaluate_model(self, history, X_test, y_test, model)	Функція, яка оцінює модель на тестових даних. Вона обчислює точність моделі (accuracy) та відображає графіки точності і втрати моделі під час навчання. Вона також обчислює матрицю помилок (confusion matrix) для прогнозованих значень та відображає її у вигляді графіка.
plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix', cmap=plt.cm.Blues)	Статичний метод класу, який відображає матрицю помилок (confusion matrix) у вигляді графіка. Він приймає матрицю помилок (cm), список класів (classes), флаг нормалізації (normalize), заголовок графіка (title) і кольорову мапу (cmap) як аргументи.

Крім того, клас «Interface» також має методи для завантаження навченої моделі, відображення графіку сигналу ЕКГ та виділення піків на графіку. Цей клас дозволяє користувачеві легко взаємодіяти з програмою для класифікації сигналів

ЕКГ та отримувати результати швидко і зручно. Функції цього класу описані у таблиці 3.3.

Таблиця 3.3 – Функції класу «Interface»

Функції	Опис
__init__(self)	Конструктор класу Interface. Ініціалізує графічний інтерфейс та викликає функцію load_model() для завантаження навченої моделі.
load_model(self)	Завантажує навчену модель, яка зберігається в файлі “best_model.h5”.
select_data(self)	Відкриває діалогове вікно для вибору файлу з даними ECG. Зчитує дані з обраного файлу, витягує назву запису та перевіряє наявність відповідного файлу заголовка. Потім відображає сигнал ECG на графіку.
plot_signal(self)	Очищає графічний канвас та відображає сигнал ECG на ньому. Сигнал нормалізується та масштабується для візуалізації.
classify_signal(self)	Класифікує обраний сигнал ECG за допомогою навченої моделі. Виконує попередню обробку даних (якщо потрібно) та здійснює прогноз на основі моделі. Результат класифікації відображається на мітці результату.
find_peaks(self)	Знаходить піки на обраному сигналі ECG за допомогою функції find_peaks з бібліотеки scipy.signal. Виділяє знайдені піки на графіку сигналу.

У даному коді використовуються різні змінні для зберігання та обробки даних, керування графічним інтерфейсом та виконання обчислень. Основні змінні подані у таблиці 3.4. У контексті поліпшення візуалізації структури системи було розроблено діаграму класів (див. рисунок 3.1), яка дозволяє краще розуміти організацію системи. Дана діаграма надає загальний огляд класів та їх взаємозв’язків, що допомагає в аналізі та проектуванні системи.

Клас “loaddata” містить імпорт необхідних бібліотек та визначення класу “loaddata” з різними функціями для завантаження та обробки даних. Ці функції виконують різні операції, пов’язані з обробкою та візуалізацією даних. Крім того,

вона використовує пакети такі як os, numpy, pandas, seaborn, matplotlib.pyplot та відповідні модулі з бібліотек sklearn та keras. Ці бібліотеки надають необхідні функціональні можливості для обробки та аналізу даних.

Таблиця 3.4 – Змінні класу «Interface»

Змінні	Опис
root	Об’єкт головного вікна інтерфейсу.
btn_select_data, btn_classify, btn_find_peaks, btn_exit	Об’єкти кнопок інтерфейсу для вибору даних, класифікації, пошуку піків та виходу з програми відповідно.
canvas	Об’єкт канвасу, на якому відображається графік сигналу ECG та виділені піки.
result_label	Об’єкт мітки, яка відображає результат класифікації сигналу ECG.
model	Завантажена навчена модель, яка використовується для класифікації сигналу ECG.
data	Масив, що містить сигнал ECG, обраний користувачем.
class_labels	Список міток класів, які використовуються для класифікації сигналу ECG.

Кожна функція в класі “loaddata” має свою роль в завантаженні та обробці даних. Наприклад, функція “import os” імпортує модуль os, який надає функціональність для взаємодії з операційною системою. Функція “import numpy as np” імпортує модуль numpy під псевдонімом np, який є потужною бібліотекою для маніпулювання масивами та обчислень науки про дані. Аналогічно, інші функції виконують імпорт необхідних модулів та функцій для виконання конкретних завдань.

Клас “ModelTraining” виступає як обгортка для тренування та оцінки моделі нейронної мережі. Він містить функції для додавання гаусівого шуму до даних, визначення архітектури моделі, навчання моделі на тренувальних даних та

оцінки її продуктивності на тестових даних. Крім того, цей клас надає можливість візуалізації графіків точності та втрат моделі під час навчання, а також графіку матриці помилок. Ці функціональні можливості сприяють аналізу та оцінці продуктивності моделі.

Клас “Interface” виконує роль графічного інтерфейсу користувача для класифікації сигналів ЕКГ. Він надає функції для вибору даних ЕКГ, класифікації сигналу, пошуку піків та відображення результатів. Крім того, цей клас дозволяє завантаження навченої моделі, відображення графіку сигналу ЕКГ та виділення піків на графіку. Інтерфейс забезпечує зручну взаємодію користувача з системою та відображення результатів класифікації.

Загалом, структура системи є добре організованою і забезпечує відповідні функціональні можливості для завантаження, обробки та класифікації сигналів ЕКГ. Використання діаграми класів та таблиць з функціями сприяє легкому розумінню структури системи та взаємозв’язків між її компонентами.

3.2 Особливості реалізації програмних складових інформаційної системи

Для балансу набору даних було необхідно створити код, що використовується для вирівнювання незбалансованих класів у DataFrame з назвою train_df. Функціональність коду наведена у послідовному порядку:

Код перетворює значення у стовпці 187 змінної train_df на цілі числа.

Потім обчислюється розподіл значень у стовпці 187 і результат зберігається у змінній equilibre.

Код імпортує необхідний модуль для виконання перевибірки.

Створюються окремі DataFrames для кожного класу у стовпці 187:

- df_1 містить рядки, де значення у стовпці 187 дорівнює 1.
- df_2 містить рядки, де значення у стовпці 187 дорівнює 2.
- df_3 містить рядки, де значення у стовпці 187 дорівнює 3.
- df_4 містить рядки, де значення у стовпці 187 дорівнює 4.

– df_0 містить випадкову вибірку 20 000 рядків, де значення у стовпці 187 дорівнює 0.

Виконується перевибірка для кожного класу з використанням resampling:

– df_1_upsample випадковим чином вибирає рядки з df_1 з повторенням для досягнення бажаного обсягу вибірки 20 000.

– df_2_upsample випадковим чином вибирає рядки з df_2 з повторенням для досягнення бажаного обсягу вибірки 20 000.

– df_3_upsample випадковим чином вибирає рядки з df_3 з повторенням для досягнення бажаного обсягу вибірки 20 000.

– df_4_upsample випадковим чином вибирає рядки з df_4 з повторенням для досягнення бажаного обсягу вибірки 20 000.

Перевибрані DataFrames об’єднуються разом для створення нового train_df, де кожен клас представлений однаковою кількістю 20 000 рядків.

Мета цієї перевибірки полягає у вирівнюванні розподілу класів у train_df, шляхом збільшення кількості зразків для менших класів (1, 2, 3 і 4), щоб вони відповідали більшому класу (0).

Даний код проводить обробку та візуалізацію даних. Він включає в себе групування даних, відбір випадкового зразка, побудову графіку та гістограми на основі підмножини даних.

Перша частина. У цій частині виконується групування датафрейму train_df за значенням у стовпці з індексом 187. Групування здійснюється методом groupby. Далі застосовується функція sample(1) до кожної групи, що повертає випадковий зразок розміром 1 з кожної групи. В результаті отримуємо новий датафрейм c, який містить по одному випадковому зразку з кожної групи.

Використовується бібліотека matplotlib.pyplot для побудови графіку. Викликається функція plot, яка будує графік на основі даних з датафрейму c. Конкретно, будується графік для рядка з індексом 0 та стовпцями від 0 до 185 (включно). Графік зображено на рисунку 3.2.

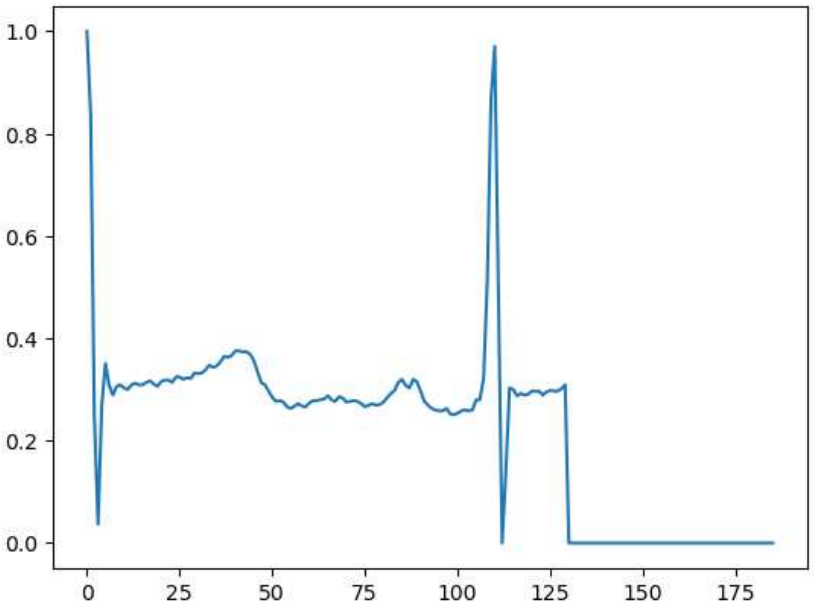


Рисунок 3.2 – Графік окреслення ЕКГ сигналу

Друга частина. У цій частині визначається функція plot_hist, яка використовується для побудови гістограми двох змінних. Функція приймає параметри class_number, size, min_ та bins.

Останній рядок plot_hist(0, 70, 5, 65) викликає функцію plot_hist з певними параметрами для побудови гістограми (рисунок 3.3).

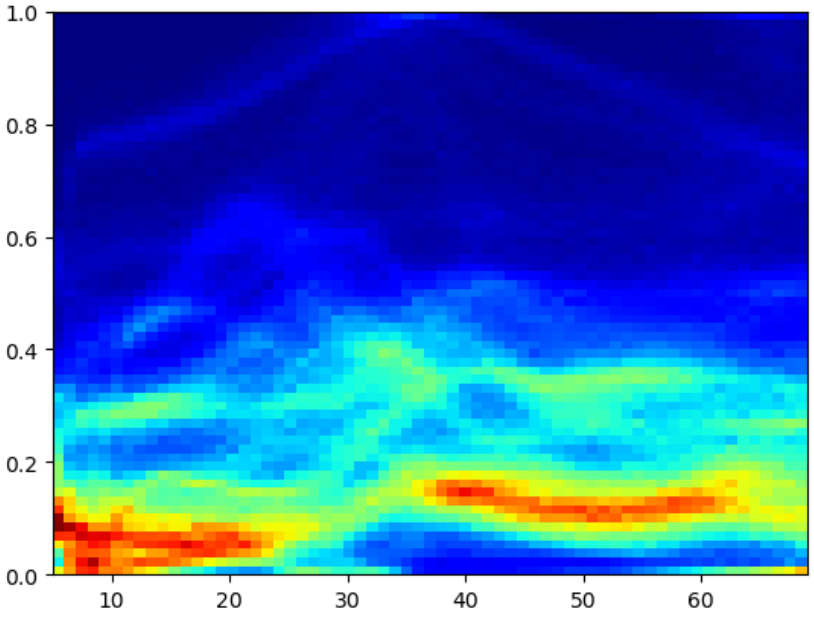


Рисунок 3.3 – Гістограма

Даний код реалізує нейромережу на основі згорткових шарів для класифікації даних. Мережа складається з послідовності шарів, які виконують операції згортки, пулінгу та повнозв'язного відображення для визначення значущих особливостей у вхідних зображеннях. Опис моделі:

im_shape=(X_train.shape[1],1) – Визначаємо розмір вхідних зображень (дані X_train) у вигляді кортежу (розмір_зображення, 1). Ось тут X_train.shape[1] повертає розмір другого розміру (тобто кількість стовпців) вхідних даних X_train.

inputs_cnn=Input(shape=(im_shape), name='inputs_cnn') – Створюється вхідний шар моделі з вказаною формою (shape) вхідних даних. im_shape використовується тут для визначення форми вхідних даних. Шар також отримує ім'я 'inputs_cnn'.

Подальші рядки коду визначають послідовність згорткових (Convolutional) та пулінгових (MaxPooling) шарів, які згорнуть та виберуть важливі особливості з вхідних зображень. Наприклад:

– conv1_1=Convolution1D(64,(6),activation='relu',input_shape=im_shap)(inputs_cnn) – Створюється перший згортковий шар з 64 фільтрами розміром (6, 1), функцією активації ReLU та вказаною формою вхідних даних.

– conv1_1=BatchNormalization()(conv1_1) – Застосовується шар нормалізації пакетів (Batch Normalization) до виходу першого згорткового шару.

– pool1=MaxPool1D(pool_size=(3),strides=(2),padding="same")(conv1_1) – Створюється перший пулінговий шар з розміром пулінгу (3) та зсувом (strides) (2), що зменшує розмір зображення в два рази.

– Аналогічно, створюються другий та третій згорткові та пулінгові шари з використанням інших параметрів.

flatten=Flatten()(pool3) – Після останнього пулінгового шару застосовується шар розгортання (Flatten), який перетворює двовимірний вихід пулінгу у вектор фіксованої довжини.

dense_end1=Dense(64,activation='relu')(flatten) та dense_end2=Dense(32, activation='relu')(dense_end1) – Додаються повнозв'язні шари (Dense) з 64 та 32 нейронами відповідно, з функцією активації ReLU.

main_output=Dense(5,activation='softmax',name='main_output')(dense_end2) – Вихідний повнозв'язний шар з 5 нейронами (відповідає кількості класів для класифікації), з функцією активації softmax для отримання ймовірностей приналежності до кожного класу.

model=Model(inputs=inputs_cnn,outputs=main_output) – Створюється модель, яка використовує визначені вхідний та вихідний шари.

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']) – Компілюється модель, вказуючи оптимізатор (adam), функцію втрати (categorical_crossentropy) та метрики, які будуть обчислюватись під час тренування (точність).

callbacks – Список зворотніх викликів, які будуть виконані під час тренування моделі. У даному випадку використовуються EarlyStopping та ModelCheckpoint. EarlyStopping зупиняє тренування, якщо втрата на перевірочних даних не зменшується впродовж 8 епох. ModelCheckpoint зберігає найкращу модель (з найменшою втратою на перевірочних даних) у файл best_model.h5.

history=model.fit(X_train,y_train,epochs=40,callbacks=callbacks,batch_size=32,validation_data=(X_test,y_test)) – Тренується модель на навчальних даних (X_train, y_train) протягом 40 епох, використовуючи зворотні виклики та пакетний розмір 32. Валідаційні дані (X_test, y_test) використовуються для оцінки моделі під час тренування.

model.load_weights(best_model.h5) – Завантажуються ваги найкращої моделі з файлу best_model.h5.

Структура мережі починається з визначення форми вхідних зображень та створення відповідного вхідного шару. Далі, чергові згорткові шари виконують операції згортки над вхідними даними, використовуючи фільтри з активацією ReLU для виділення значущих ознак. Після кожного згорткового шару використовується шар нормалізації пакетів для стабілізації та прискорення тренування.

Після згорткових шарів використовується пулінговий шар, який зменшує розмір зображення та підсилює найважливіші особливості. Після трьох

послідовних згорткових та пулінгових шарів отримані вихідні дані розгортаються у вектор за допомогою шару розгортання (Flatten).

Наступні два повнозв’язні шари (Dense) приймають отриманий вектор та виконують нелінійні перетворення з активацією ReLU. Остаточний повнозв’язний шар має кількість нейронів, що відповідає кількості класів для класифікації. Функція активації softmax застосовується для отримання ймовірностей належності до кожного класу.

Модель компілюється з використанням оптимізатора Adam та функції втрати категоріальної перехресної ентропії. Під час тренування також обчислюється метрика точності. Для контролю процесу тренування використовуються два зворотні виклики: EarlyStopping та ModelCheckpoint. EarlyStopping зупиняє тренування, якщо втрата на перевірочних даних не зменшується протягом заданої кількості епох. ModelCheckpoint зберігає ваги моделі з найкращою втратою на перевірочних даних.

Мережа тренується на навчальних даних з використанням зазначених гіперпараметрів, таких як кількість епох та розмір пакета. Валідаційні дані використовуються для оцінки моделі під час тренування. Після завершення тренування найкращі ваги моделі завантажуються з файлу best_model.h5.

У підсумку, даний код реалізує нейромережу CNN зі згортковими шарами для класифікації даних та забезпечує ефективне тренування моделі з використанням зворотних викликів та збереженням найкращих параметрів моделі.

Основні функції, що реалізують процес навчання нейронно мережі наведено у таблиці 3.5.

Таблиця 3.5 – Функції «evaluate_model»

Функції	Опис
scores = model.evaluate((X_test), y_test, verbose=0)	Модель оцінюється на тестових даних X_test та y_test, і результати оцінки зберігаються у змінній scores.

print(“Accuracy: %.2f%%” % (scores[1]*100))	Виводиться точність моделі у відсотках.
print(history)	Виводиться історія навчання моделі.
plt.plot(history.history[‘accuracy’]) та plt.plot(history.history[‘val_accuracy’])	Побудова графіка точності моделі під час навчання на тренувальних та валідаційних даних.
plt.plot(history.history[‘loss’]) та plt.plot(history.history[‘val_loss’])	Побудова графіка втрат моделі під час навчання на тренувальних та валідаційних даних.
y_true=[] та for element in y_test: y_true.append(np.argmax(element))	Створення списку y_true, який містить індекси класів замість векторного представлення міток y_test.
prediction_proba=model.predict(X_test)	Передбачення ймовірностей для класів на тестових даних X_test за допомогою моделі.
prediction=np.argmax(prediction_proba,axis=1)	Вибір класу з найвищою ймовірністю для кожного прикладу у prediction_proba.
cnf_matrix = confusion_matrix(y_true, prediction)	Обчислення матриці помилок (confusion matrix) з використанням істинних міток y_true та передбачених міток prediction.

Наприклад, у таблиці описано функцію evaluate_model, яка використовується для оцінки моделі машинного навчання. Цей код дозволяє оцінити модель машинного навчання, вивести її точність, побудувати графіки точності та втрат під час навчання, а також отримати матрицю помилок для оцінки результатів класифікації моделі. Функція plot_confusion_matrix використовується для візуалізації матриці плутанини (confusion matrix) в задачах класифікації. Матриця плутанини показує кількість правильно та неправильно класифікованих зразків для кожного класу. Результат матриці зображено на рисунку 3.4.

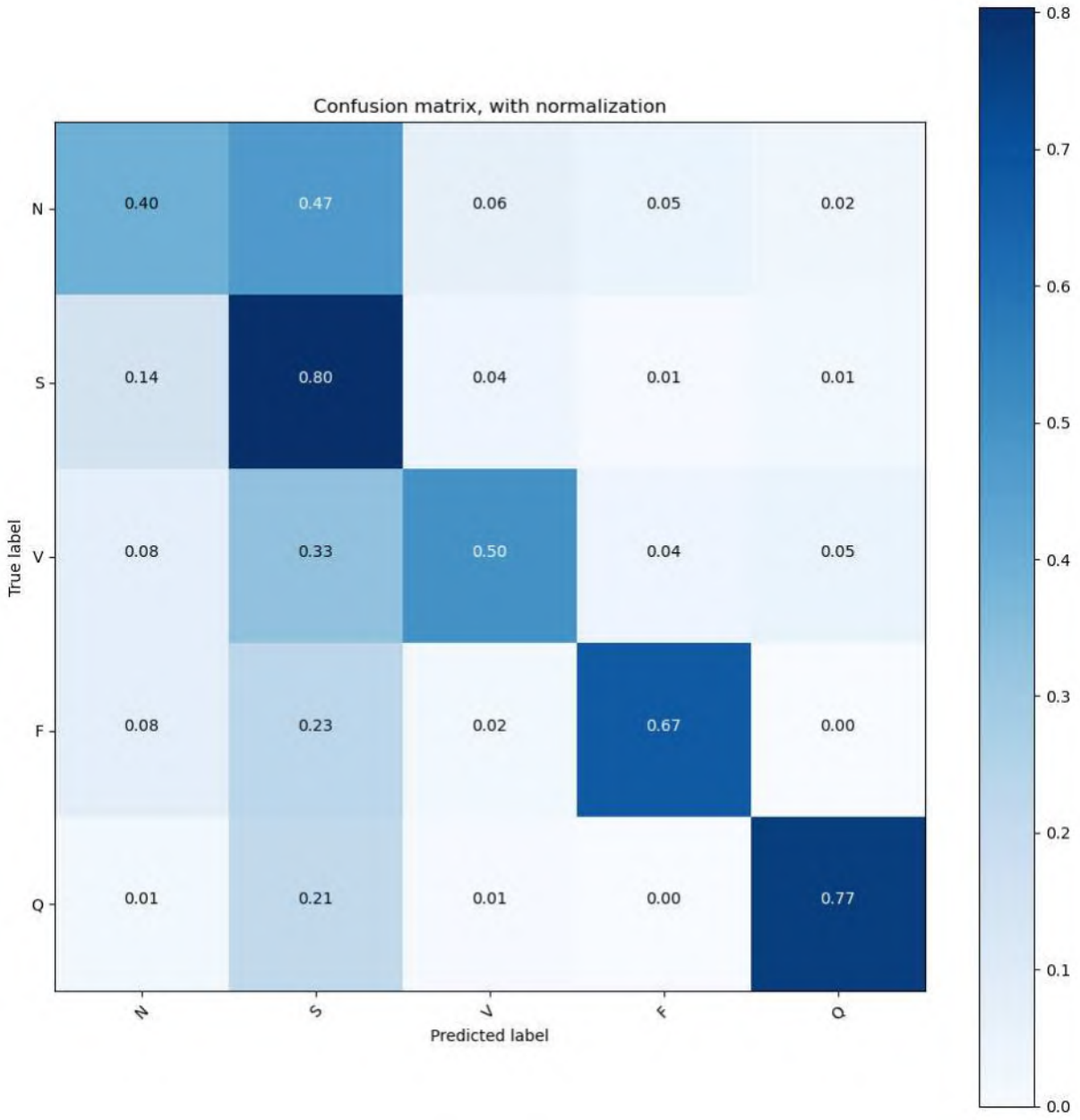


Рисунок 3.4 – Матриця невизначеності, що демонструє результати класифікації.

Функція plot_confusion_matrix приймає такі аргументи:

– cm: матриця плутанини, яку потрібно візуалізувати.

– classes: список класів, які відповідають класифікації.

У коді також проводиться обчислення матриці плутанини за допомогою функції confusion_matrix. Після цього встановлюються опції виводу чисел з плаваючою комою за допомогою np.set_printoptions(precision=2).

Нарешті, створюється фігура для візуалізації матриці плутанини без нормалізації за допомогою plt.figure(figsize=(10, 10)), та викликається функція plot_confusion_matrix для відображення матриці плутанини.

Матриця плутанини обчислюється за допомогою функції confusion_matrix, якій передаються два аргументи: y_test.argmax(axis=1) – це реальні мітки класів тестового набору даних, а y_pred.argmax(axis=1) – це передбачені мітки класів моделі. argmax(axis=1) використовується для отримання індексів найбільш ймовірних класів з масиву передбачень.

Потім, за допомогою функції plot_confusion_matrix, матриця плутанини візуалізується на графіку. Для цього використовуються різні параметри, такі як назва графіку (title), підписи осей (xlabel та ylabel), список класів (classes) і кольорова мапа (cmap).

У даному коді також встановлюється опція виводу чисел з плаваючою комою за допомогою np.set_printoptions(precision=2), щоб відобразити значення в матриці плутанини з двома десятковими знаками.

3.3 Експериментальне тестування інформаційної системи

Експериментальне тестування інформаційної системи є важливим етапом розробки програмного забезпечення, яке дозволяє перевірити його функціональність, ефективність та надійність перед його впровадженням. Цей процес включає проведення спеціальних експериментів, під час яких систему піддають різним сценаріям та навантаженням з метою виявлення потенційних проблем і помилок.

Один із методів тестування буде використання тест-кейсів. Тест-кейси є важливою складовою експериментального тестування інформаційної системи. Вони представляють собою детально описані кроки тестування, які дозволяють виконати певні функції або сценарії системи з метою перевірки їх правильності та відповідності вимогам.

Можна провести тестування на помилки, якщо дані не були обрані перед виконанням певних операцій, наприклад, натискання кнопок “Вибрати дані”, “Класифікувати” або “Знайти пікові точки” без вибраних даних (таблиця 3.6).

Таблиця 3.6 – Тест-кейс WTJ001

<table>
<tr><td>Тест-кейс ID: WTJ001</td><td>Пріоритет: 2</td><td>Створено: 20.05.2023,
Федорчук І. І.</td></tr>
<tr><td colspan="3">Назва: Тестування на помилку, якщо дані не були обрані перед виконанням певних операцій, наприклад, натискання кнопок “Вибрати дані”.
Вхідні дані: Натискання на кнопку «Вибрати дані» та вибір файла.</td></tr>
<tr><td>Кроки</td><td colspan="2">Очікуваний результат</td></tr>
<tr><td>Передумова: користувач вибрав дані, що не мають необхідні числа, символи
1. Запустити програму
2. Натиснути кнопку «Вибрати дані»
3. Вибрати файл формата .dat</td><td colspan="2">З’являється вікно, яке сповіщує користувача про те, що «не вибрано жодних даних»</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Вікно, що повідомляє про помилку, зображено на рисунку 3.5.

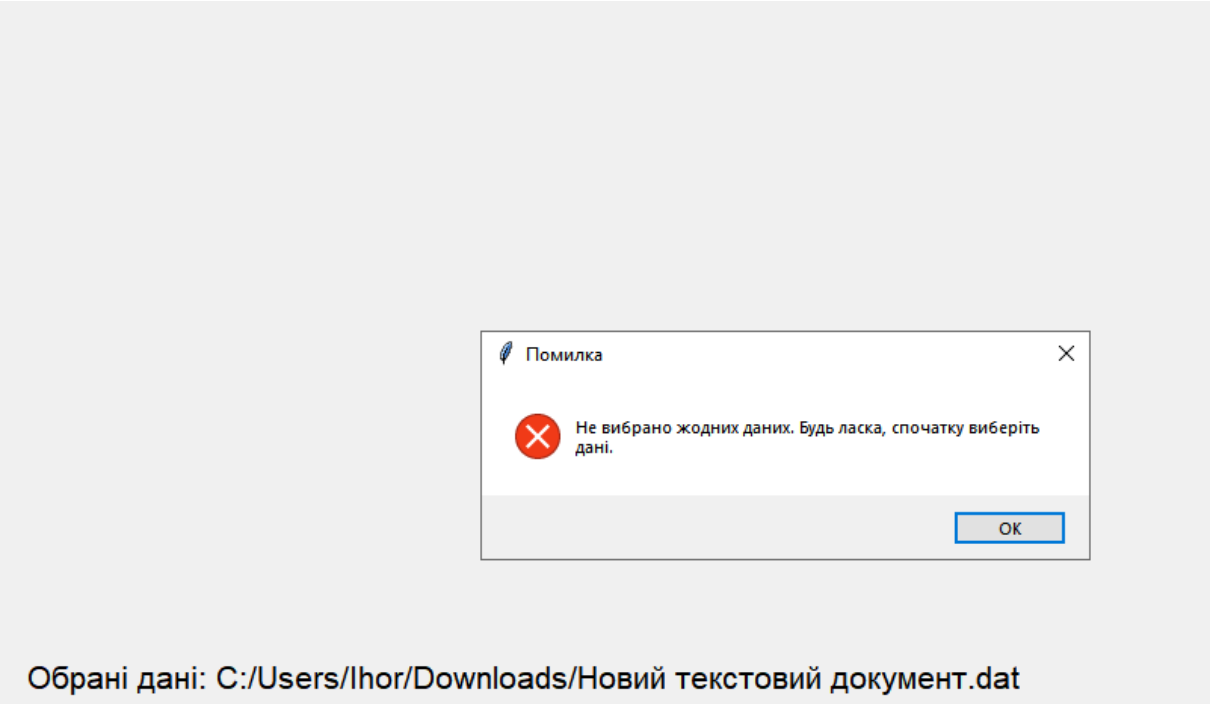


Рисунок 3.5 – Демонстрація сповіщення помилки

Опис тест-кейсу WTJ002 подано в таблиці 3.7.

Таблиця 3.7 – Тест-кейс WTJ002

<table>
<tr><td>Тест-кейс ID: WTJ002</td><td>Пріоритет: 2</td><td>Створено: 20.05.2023,
Федорчук І. І.</td></tr>
<tr><td colspan="3">Назва: Тестування на помилку, якщо дані не були обрані перед виконанням певних операцій, наприклад, натискання кнопок «Класифікувати».
Вхідні дані: Натискання на кнопку «Класифікувати».</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">Передумова: користувач вибрав дані, що не мають необхідні числа, символи
1. Запустити програму
2. Натиснути кнопку «Класифікувати»</td><td>З’являється вікно, яке сповіщує користувача про те, що «не вибрано жодних даних»</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Після натискання кнопки «Класифікувати» без завантажених даних, програма буде сповіщати користувача про те, що не був встановлений файл з відповідними даними. Отже, програма буде сповіщати про помилку. Демонстрація сповіщення помилки зображено на рисунку 3.6.

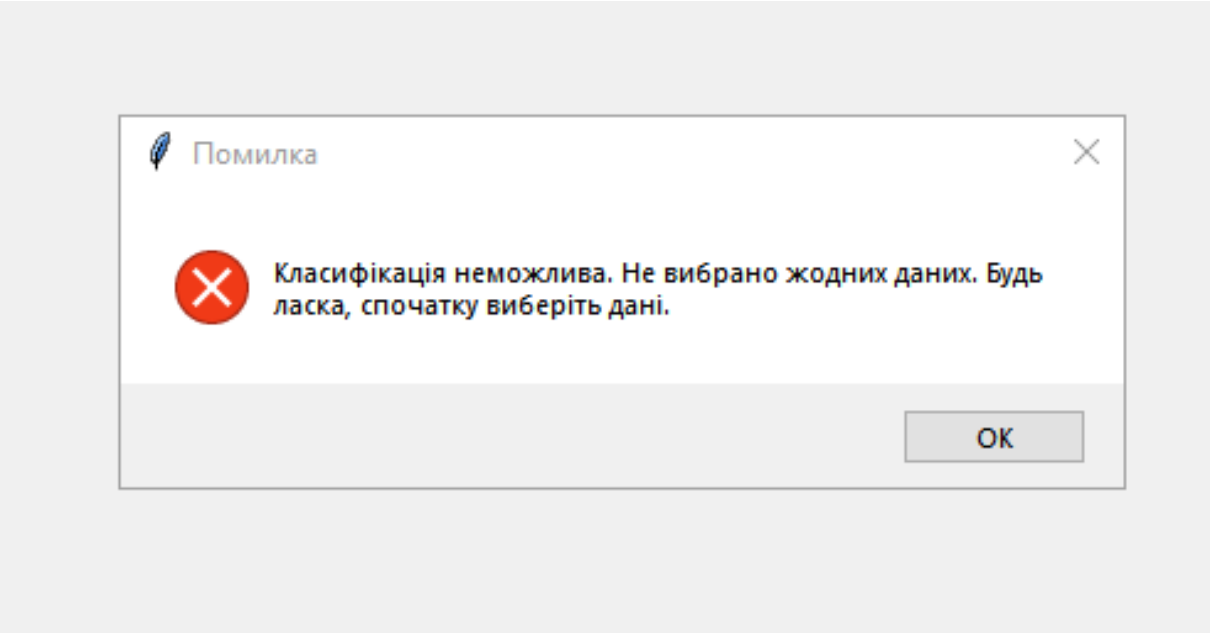


Рисунок 3.6 – Сповіщення про помилку класифікації

Опис тест-кейсу WTJ003 в таблиці 3.8.

Таблиця 3.8 – Тест-кейс WTJ003

Тест-кейс ID: WTJ003	Пріоритет: 2	Створено: 20.05.2023, Федорчук І. І.
Назва: Тестування на помилку, якщо дані не були обрані перед виконанням певних операцій, наприклад, натискання кнопок «Знайти пікові точки». Вхідні дані: Натискання на кнопку «Знайти пікові точки».		
Кроки		Очікуваний результат
Передумова: користувач вибрав дані, що не мають необхідні числа, символи 1. Запустити програму 2. Натиснути кнопку «Знайти пікові зубці QR»		З'являється вікно, яке сповіщує користувача про те, що «не вибрано жодних даних»
Результат виконання тест-кейсу: пройдено успішно		

Після натискання кнопки «Знайти пікові точки» без завантажених даних, програма буде сповіщати користувача, що файл з даними не був завантажений. Демонстрація сповіщення також зображено на рисунку 3.7.

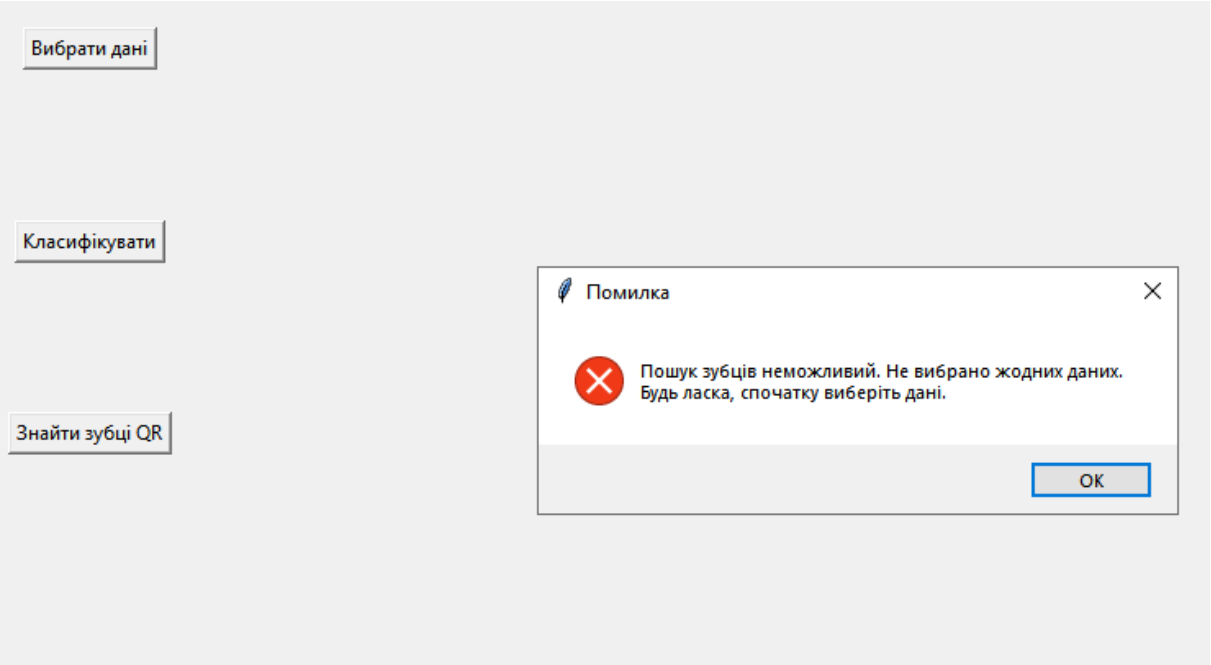


Рисунок 3.7 – Демонстрація сповіщення помилки у пошуках зубців

Опис тест-кейсу WTJ004 в таблиці 3.9.

Таблиця 3.9 – Тест-кейс WTJ004

Тест-кейс ID: WTJ004	Пріоритет: 1	Створено: 20.05.2023, Федорчук І. І.
Назва: Тестування на успішне завантаження даних. Вхідні дані: Натискання на кнопку «Вибрати дані».		
Кроки	Очікуваний результат	
Передумова: Завантаження даних 1. Запустити програму 2. Натиснути кнопку «Вибрати дані» 3. Вибрати файл формату .dat з директорії MIT-BIH Arrhythmia Database 4. Завантажити	Після певного часу обробки програмою файлу на графічному інтерфейсі з'явиться вибраний сигнал ЕКГ.	
Результат виконання тест-кейсу: пройдено успішно		

Отже, після успішного завантаження файлу з даними на графічному інтерфейсі відобразиться ЕКГ-сигнал. Завантажений файл буде відображатися на інтерфейсі, де буде вказувати шлях до директорії, з якої він був завантажений. Результат тестування зображено на рисунку 3.8.

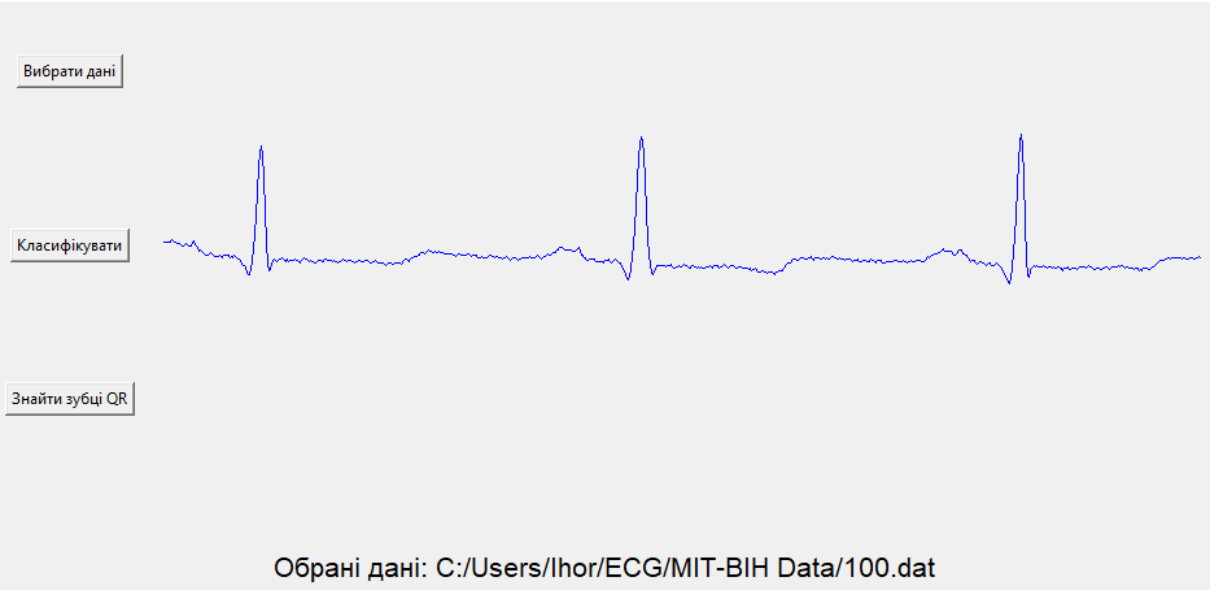


Рисунок 3.8 – Результат успішної роботи функції «Вибрати дані»

Опис тест-кейсу WTJ005 в таблиці 3.10.

Таблиця 3.10 – Тест-кейс WTJ005

Тест-кейс ID: WTJ005	Пріоритет: 1	Створено: 20.05.2023, Федорчук І. І.
Назва: Тестування на успішне окреслення ЕКГ-сигналу Вхідні дані: Натискання на кнопки «Вибрати дані», «Класифікація» та «Знайти зубці QR»		
Кроки		Очікуваний результат
Передумова: Опис ЕКГ-сигналу 1. Запустити програму 2. Натиснути кнопку «Вибрати дані» 3. Вибрати файл формату .dat з директорії MIT-BIH Arrhythmia Database 4. Завантажити 5. Натиснути кнопку «Класифікація» 6. Натиснути кнопку «Знайти зубці QR»		На графічному інтерфейсі будуть відображатися клас ЕКГ-сигналу, а також зубці Q та R. Додатковою інформацією будуть назва запису, кількість каналів, частота дискретизації та види каналів і т.д.
Результат виконання тест-кейсу: пройдено успішно		

Необхідна інформація про ЕКГ-сигнал відображається на графічному інтерфейсі інформаційної системи. Результат окреслення та класифікації сигналу зображено на рисунку 3.9.

Рисунок 3.9 – Опис ЕКГ-сигналу

Під час експериментального тестування інформаційної системи було використано тест-кейс для перевірки сповіщення про помилку не завантаження даних, успішне завантаження та успішний опис ЕКГ-сигналу.

Тест-кейс містив послідовність дій, які спрямовані на спровокування ситуації, коли дані не вдалося завантажити.

Під час тестування системи за допомогою цього тест-кейсу, виявлено, що система правильно сповіщає про помилку. Таке повідомлення дозволяє користувачам вчасно реагувати на проблему та здійснювати необхідні кроки для вирішення цієї проблеми.

Використання тест-кейсу на перевірку сповіщення про помилку дозволяє переконатися, що система вміє ефективно реагувати на такі ситуації та забезпечує користувачам важливу інформацію для подальшого вирішення проблеми.

Перевірка успішності виконання може бути здійснена за допомогою тест-кейсу, що спрямований на перевірку праці функцій інформаційної системи. Цей тест-кейс дозволяє впевнитися, що система адекватно реагує і надає користувачам необхідну інформацію.

3.4 Вимоги до розгортання інформаційної системи та інструкція користувача

Отже, для досягнення бажаного результату, а саме: завантаження ЕКГ-сигналу, класифікації ЕКГ-сигналу та його візуалізації, необхідно запустити програму. Після успішного запуску програми користувач матиме можливість взаємодіяти з графічним інтерфейсом. На рисунку 3.10 зображено зображення графічного інтерфейсу програми.

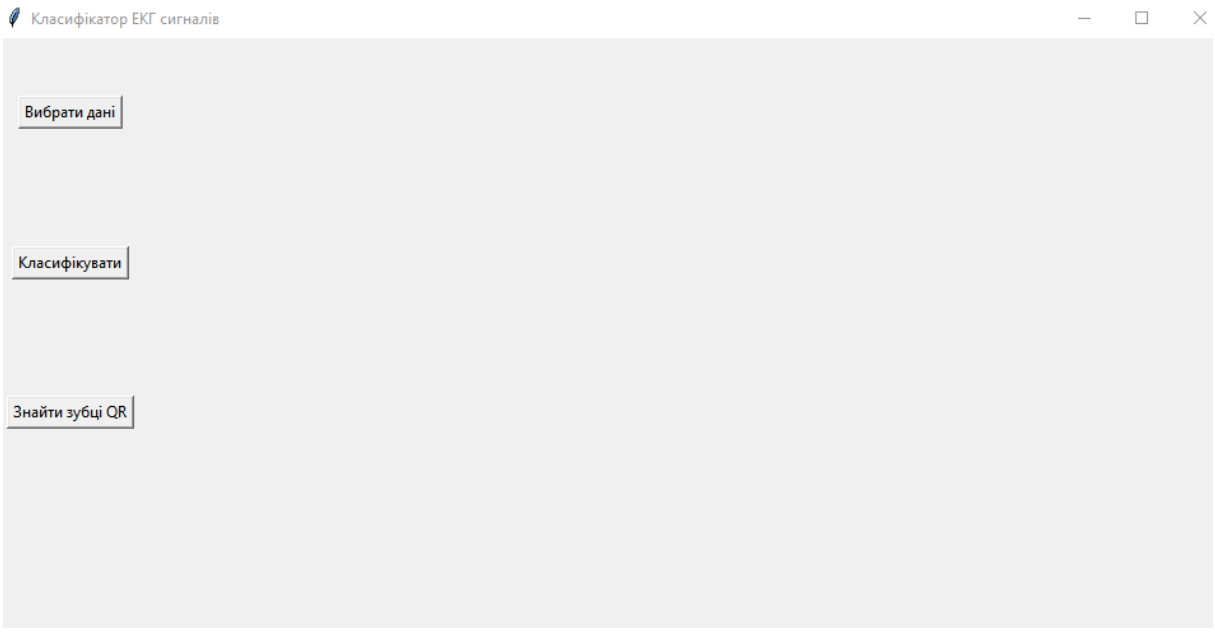


Рисунок 3.10 – Графічний інтерфейс програмного застосунку

Після ознайомлення з інтерфейсом користувачу необхідно завантажити дані формату .dat з директорії бази даних Mit-Bih Arrhythmia Database, яка встановлена на його персональному комп’ютері, або встановити її, якщо вона не встановлена. Провідник з необхідними файлами зображено на рисунку 3.11.

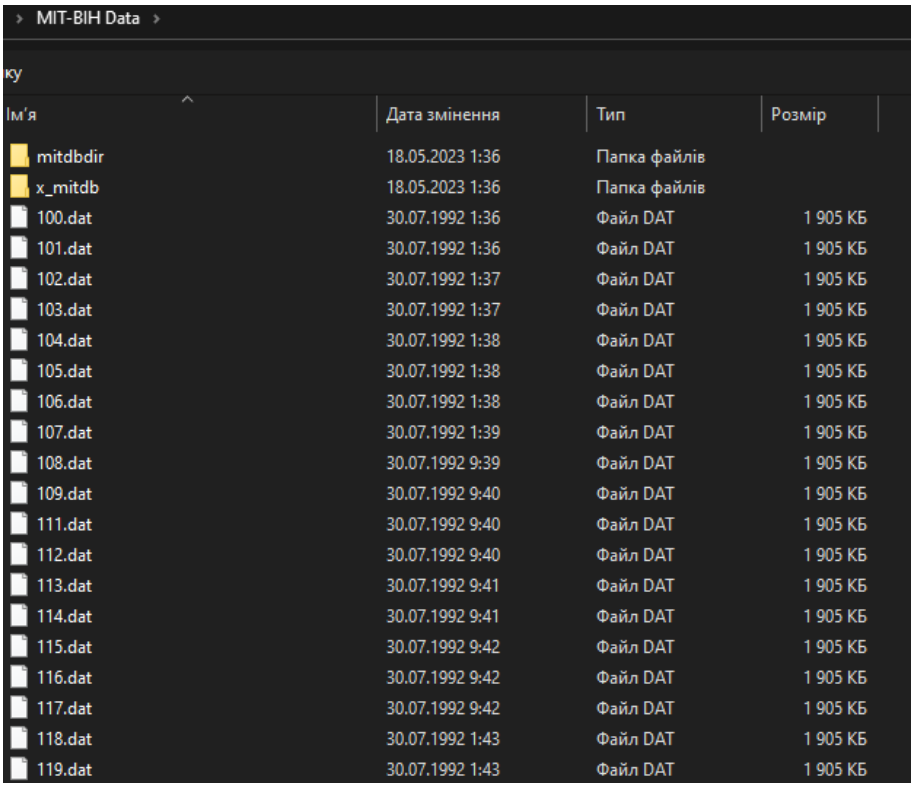


Рисунок 3.11 – Директорія бази даних Mit-Bih Arrhythmia Database

Після завершення цієї операції, на графічному інтерфейсі з’явиться ЕКГ-сигнал разом з додатковою інформацією про нього. Демонстрація ЕКГ-сигналу на графічному інтерфейсі зображено на рисунку 3.12.

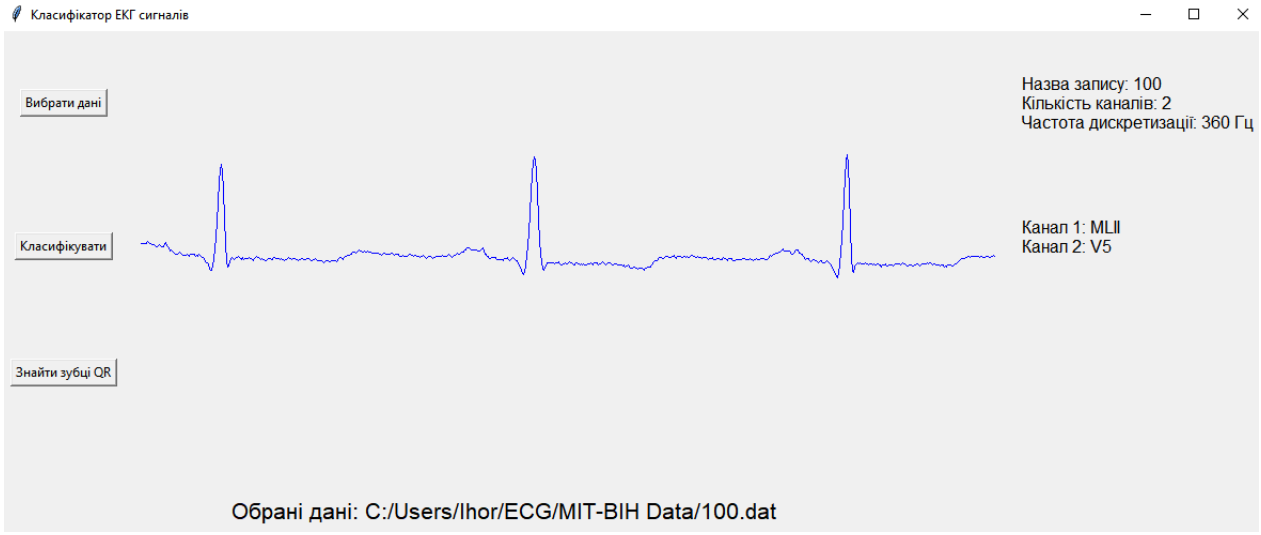


Рисунок 3.12 – Демонстрація ЕКГ-сигналу на графічному інтерфейсі

Далі користувач може натиснути кнопку “Класифікувати”. Програма спробує вірно класифікувати вибраний користувачем ЕКГ-сигнал. Реалізацію цієї функції можна побачити на рисунку 3.13.

Рисунок 3.13 – Демонстрація результату класифікації ЕКГ-сигналу

На рисунку 3.12 зображено результат “N”, що означає «Non-ectopic beats» (нормальний ритм серця). Програма загалом вміє класифікувати кілька типів ЕКГ-сигналів, зокрема:

– «N»: Non-ectopic beats (нормальний ритм серця).

– «S»: Supraventricular ectopic beats (суправентрикулярні ектопічні ритми).

– «V»: Ventricular ectopic beats (шлунково-шлуночкові ектопічні ритми).

– «F»: Fusion Beats (змішані ритми).

– «Q»: Unknown Beats (невідомий тип ритму).

Після класифікації ЕКГ-сигналу, користувач може натиснути на кнопку “Знайти зубці QR”. Після цього на ЕКГ-сигналі будуть видні точки Q та R. Результат цієї операції зображений на рисунку 3.14.

Рисунок 3.14 – Демонстрація результату кнопки «Знайти зубці QR»

У програмному застосунку розроблено графічний інтерфейс, який дозволяє користувачеві взаємодіяти з програмою.

Щоб забезпечити належну функціональність цієї інформаційної системи, необхідно відповідати конкретним вимогам:

– Windows 10;

– Google Chrome;

– Python 3.11.3;

– Jupyter Notebook;

– Anaconda Navigator.

Отже, програма забезпечує користувачеві зручний спосіб завантаження, класифікації та візуалізації ЕКГ-сигналу, а також знаходження точок Q та R. Це може бути корисним інструментом для аналізу ритму серця та виявлення аномалій у ЕКГ-сигналах.

3.5 Висновки до розділу 3

У розділі 3 наведено опис розробку та реалізації комплексної системи, яка забезпечує завантаження, обробку, класифікацію та візуалізацію ЕКГ-сигналів з метою виявлення аномалій та аналізу ритму серця.

Створено структуру системи, яка дозволяє краще розуміти організацію компонентів та їх взаємозв’язки. Для цього було розроблено діаграму класів, яка надає загальний огляд класів та їх функціональності. Система містить реалізовану нейромережеву модель, використовуючи згорткові та пулінгові шари для зменшення розміру зображення та підсилення особливостей. Модель має повнозв’язні шари з активацією ReLU та функцію активації softmax для отримання ймовірностей класифікації.

Система також має можливості сповіщення про помилки, що допомагають користувачу зрозуміти, які проблеми виникли під час роботи з системою та як їх вирішити. Наприклад, система може повідомляти про помилки завантаження даних або про незавершене тренування моделі.

Ця програмна система може бути використана для проведення досліджень у галузі аналізу ритму серця та виявлення аномалій у ЕКГ-сигналах. Користувачі можуть завантажувати свої ЕКГ-сигнали, обробляти їх, класифікувати та отримувати результати аналізу через інтуїтивний графічний інтерфейс.

Висновки

В результаті виконання кваліфікаційної роботи бакалавра була досягнута поставлена мета – покращено окреслення сигналів електрокардіограм. Для досягнення цієї мети були виконані наступні завдання.

Початково був проведений детальний аналіз різних методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм з метою вибору найкращого підходу. Цей аналіз надав змогу визначити оптимальний метод опису та формалізації сигналів електрокардіограм.

Після вибору найбільш підходящого способу цифрового опису та формалізації сигналів електрокардіограм, його було успішно застосовано для розв'язання задачі автоматизованого окреслення сигналів електрокардіограм. Це дозволило досягти точного та автоматизованого окреслення сигналів електрокардіограм з високою надійністю.

Далі, обраний метод було реалізовано у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм. Розроблений модуль демонструє ефективну роботу і забезпечує точне та швидке окреслення сигналів електрокардіограм.

Для підтвердження ефективності реалізованого модуля було проведено експериментальне тестування з використанням еталонних наборів даних. Результати експериментального тестування підтвердили високу точність і ефективність розробленого модуля програмного забезпечення, що підтверджує досягнення поставленої мети.

Отже, на основі проведеного аналізу, реалізації та експериментального тестування можна зробити узагальнюючий висновок, що розроблений модуль програмного забезпечення забезпечує високу точність окреслення сигналів електрокардіограм, що сприяє поліпшенню діагностики та аналізу серцево-судинних захворювань. Застосування цього модуля може допомогти медичному персоналу в ранньому виявленні та лікуванні серцевих захворювань, а також покращити загальний стан пацієнтів.

Розроблений модуль програмного забезпечення має потенціал для подальшого розширення та вдосконалення. Наприклад, можливо врахувати різні фактори, такі як вік пацієнта, стать, наявність певних хвороб або інших медичних показників, що можуть впливати на аналіз електрокардіограм. Таке вдосконалення може допомогти покращити точність та адаптивність модуля до індивідуальних потреб кожного пацієнта. Крім того, можна розглянути можливість інтеграції модуля з іншими системами медичного обладнання або електронними медичними записами для полегшення обміну даними та вдосконалення управління інформацією. З такими подальшими розширеннями модуль програмного забезпечення може стати ще більш цінним інструментом у медичній галузі, сприяючи точнішій діагностиці та забезпечуючи краще здоров'я пацієнтів.

Перелік посилань

1. Dey S., Pal R., Biswas S. Deep learning algorithms for efficient analysis of ECG signals to detect heart disorders. In: Asadpour V., Karakuş S. *Biomedical Engineering*. IntechOpen, 2022. Vol. 14. Pp. 1-25.

2. Krak Iu., Barmak O., Radiuk P. Information technology for early diagnosis of pneumonia on individual radiographs. *The 3rd International Conference on Informatics & Data-Driven Medicine (IDDM 2020)* : CEUR-Workshop Proceedings. Vol. 2753. (Växjö, Sweden, 19-21 November 2020). CEUR-WS.org, Aachen, 2020. Pp. 11-21.

3. Каплунова А.С. Метод та алгоритм обробки ЕКГ-сигналу в умовах невизначеності : кваліфікаційна робота магістра за спеціальністю 163 Біомедична інженерія / А.С. Каплунова. Тернопіль: ТНТУ, 2022. 54 с.

4. Основи електрокардіографії | Настанова з кардіології | Довідник лікарських препаратів Компендіум. *Компендіум*. URL: https://compendium.com.ua/uk/clinical-guidelines-uk/cardiology-uk/section-5-uk/glava-1-osnovi-elektrokardiografiyi/

5. Radiuk P., Barmak O., Krak Iu. An approach to early diagnosis of pneumonia on individual radiographs based on the CNN information technology. *The Open Bioinformatics Journal*. 2021. Vol. 14, No 1. Pp. 93-107.

6. ЕЛЕКТРОКАРДІОГРАФІЯ. *Фармацевтична енциклопедія*. URL: https://www.pharmencyclopedia.com.ua/article/2331/elektrokardiografiya

7. Бармак О.В., Радюк П.М. Інформаційна технологія візуального аналізу рентгенівських зображень для інтерпретації результатів діагностування пневмонії. *Вісник Хмельницького національного університету. Технічні науки*. 2021. № 295(2). С. 52-55.

8. Radiuk P.M., Skrypnyk T.K., Karlechuk D.T. Applying mental models to making controlled critically safe decisions in IT project management. *Herald of Khmelnytskyi National University. Technical sciences*. 2021. Vol. 301, No 5. Pp. 32-35.

9. Application of deep learning techniques for heartbeats detection using ECG signals-analysis and review / F. Murat et al. *Computers in Biology and Medicine*. 2020. Vol. 120. Pp. 103726.

10. Detection of R-peaks using fractional Fourier transform and principal component analysis / V. Gupta et al. *Journal of Ambient Intelligence and Humanized Computing*. 2021. Vol. 13. Pp. 961-972.

11. Signal classification using wavelet-based features and support vector machines – MATLAB & Simulink Example. *MathWorks – MATLAB & Simulink*. URL: https://www.mathworks.com/help/wavelet/ug/ecg-classification-using-wavelet-features.html

12. Human-in-the-loop approach based on MRI and ECG for healthcare diagnosis / P. Radiuk et al. *The 5th International Conference on Informatics & Data-Driven Medicine (IDDM-2022)* : CEUR-Workshop Proceedings. Vol. 3302. (Lyon, France, 18-20 November 2022). CEUR-WS.org, Aachen, 2022. Pp. 9-20.

13. A novel feature vector for ECG classification using deep learning / O. Kovalchuk et al. *The 4th International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2023)* : CEUR-Workshop Proceedings. Vol. 3373. (Khmelnytskyi, Ukraine, 22-24 March 2023). CEUR-WS.org, Aachen, 2023. Pp. 227-238.

14. A model for obstructive sleep apnea detection using a multi-layer feed-forward neural network based on electrocardiogram, pulse oxygen saturation, and body mass index / Z. Li et al. *Sleep and Breathing*. 2021. Vol. 25. Pp. 2065-2072.

15. Automated detection and classification of arrhythmia from ECG signals using feature-induced long short-term memory network / B. Ganguly et al. *IEEE Sensors Letters*. 2020. Vol. 4, No. 8. Pp. 1-4.

16. Singh P., Sharma A. Attention-based convolutional denoising autoencoder for two-lead ECG denoising and arrhythmia classification. *IEEE Transactions on Instrumentation and Measurement*. 2022. Vol. 71. Pp. 1-10.

17. Ясенко Л.С., Клятченко Я.М. Властивості згорткові нейронної мережі на основі автокодера. *Інформаційні технології та комп'ютерна інженерія*. № 3, 2021. С. 77-85.

18. An introduction to Tkinter. *McGill School Of Computer Science*. URL: https://www.cs.mcgill.ca/~hv/classes/MS/TkinterPres/

19. TensorFlow.js: Machine learning for the web and beyond / D. Smilkov et al. *2nd Conference on Machine Learning and Systems (SysML-2019)* : Proceedings. (Palo Alto, CA, USA, 31 March - 01 April 2019). MLSys.org, 2019. Pp. 1-13.

20. A Shallow U-Net architecture for reliably predicting blood pressure (BP) from photoplethysmogram (PPG) and electrocardiogram (ECG) signals / S. Mahmud et al. *Sensors*. 2022. Vol. 22, No. 3. Pp. 919.

21. Dense convolutional network and its application in medical image analysis / T. Zhou et al. *BioMed Research International*. 2022. Vol. 2022. Pp. 1-22.

22. Sharma P., Dinkar S.K., Gupta D.V. A novel hybrid deep learning method with cuckoo search algorithm for classification of arrhythmia disease using ECG signals. *Neural Computing and Applications*. 2021. Vol. 33. Pp. 13123-13143.

23. Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review / S. Hong et al. *Computers in Biology and Medicine*. 2020. Vol. 122. Pp. 103801.

24. Autoencoder for ECG signal outlier processing in system of biometric authentication / V.V. Khoma et al. *Artificial Intelligence*. 2019. Vol. 24, No. 1-2. Pp. 108-117.

25. Digitizing ECG image: A new method and open-source software code / J.D. Fortune et al. *Computer Methods and Programs in Biomedicine*. 2022. Vol. 221. Pp. 106890.

26. Stoyanov T. Web-based software tool for electrocardiogram annotation. *Contemporary Methods in Bioinformatics and Biomedicine and Their Applications* : Lecture Notes in Networks and Systems. Vol. 374. Springer, Cham, 2022. Pp. 322-331.

27. Реутська С.В. Математичне та програмне забезпечення обробки ЕКГ сигналів : кваліфікаційна робота магістра за спеціальністю 121 Інженерія програмного забезпечення / С.В. Реутська. Київ: КПІ ім. Ігоря Сікорського, 2022. 106 с.

28. Khalaf A.J., Mohammed S.J. Verification and comparison of MIT-BIH arrhythmia database based on number of beats. *International Journal of Electrical and Computer Engineering (IJECE)*. 2021. Vol. 11, No. 6. Pp. 4950.

29. Python 3.11 | Python is a programming language that lets you work quickly and integrate systems more effectively. *Python.org*. URL: https://www.python.org/

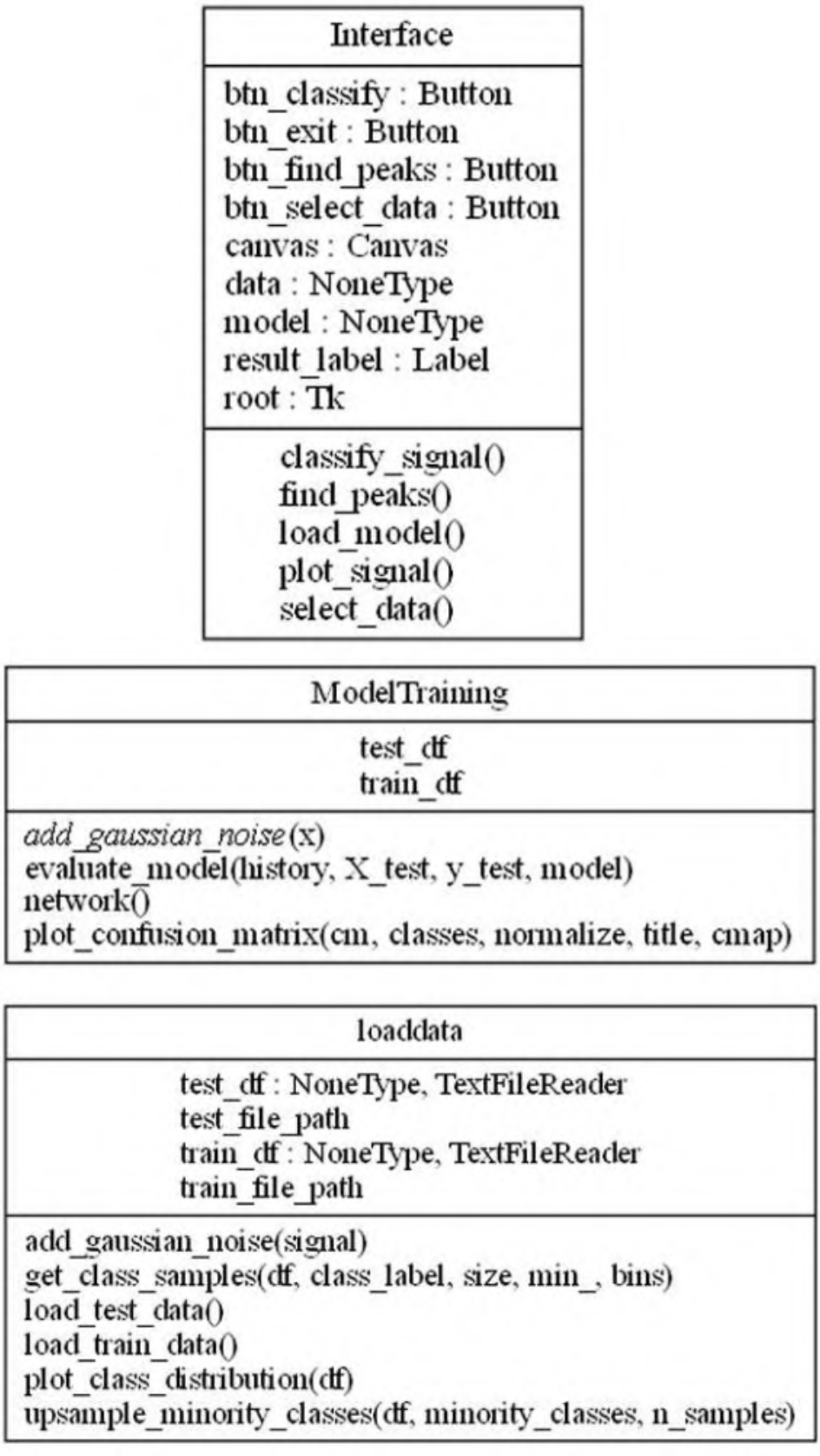
30. Python™ Package Index. Requests 2.31.0. *PyPI.org*. URL: https://pypi.org/project/requests/

31. Géron A. Hands-On machine learning with Scikit-Learn, Keras, and TensorFlow : 3rd ed. O'Reilly Media, Inc., 2022. 864 p.

ДОДАТКИ

Додаток А

Діаграма класів



Додаток Б

Лістинг програмного коду

```
import os
for dirname, _, filenames in os.walk('D:/ecg-heartbeat-categorization-dataset'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
import numpy as np
import keras
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from keras.layers import Input, Convolution1D, BatchNormalization, MaxPool1D, Flatten,
Dense
from keras.models import Model
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.utils import resample
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from sklearn.utils import class_weight
import warnings
warnings.filterwarnings('ignore')
train_df=pd.read_csv('D:/ecg-heartbeat-categorization-
dataset/mitbih_train.csv',header=None)
test_df=pd.read_csv('D:/ecg-heartbeat-categorization-
dataset/mitbih_test.csv',header=None)
train_df[187]=train_df[187].astype(int)
equilibre=train_df[187].value_counts()
print(equilibre)
from sklearn.utils import resample
df_1=train_df[train_df[187]==1]
df_2=train_df[train_df[187]==2]
df_3=train_df[train_df[187]==3]
df_4=train_df[train_df[187]==4]
df_0=(train_df[train_df[187]==0]).sample(n=20000,random_state=42)

df_1_upsample=resample(df_1,replace=True,n_samples=20000,random_state=123)
df_2_upsample=resample(df_2,replace=True,n_samples=20000,random_state=124)
df_3_upsample=resample(df_3,replace=True,n_samples=20000,random_state=125)
df_4_upsample=resample(df_4,replace=True,n_samples=20000,random_state=126)

train_df=pd.concat([df_0,df_1_upsample,df_2_upsample,df_3_upsample,df_4_upsample])
equilibre=train_df[187].value_counts()
print(equilibre)
c = train_df.groupby(187, group_keys=False).apply(lambda x: x[x[187] ==
x[187].unique()[0]].sample(1))
c
plt.plot(c.iloc[0,:186])
def plot_hist(class_number,size,min_,bins):
    img=train_df.loc[train_df[187]==class_number].values
    img=img[:,min_:size]
    img_flatten=img.flatten()

    final1=np.arange(min_,size)
    for i in range (img.shape[0]-1):
        tempo1=np.arange(min_,size)
```

```
        final1=np.concatenate((final1, tempo1), axis=None)
    print(len(final1))
    print(len(img_flatten))
    plt.hist2d(final1,img_flatten, bins=(bins,bins),cmap=plt.cm.jet)
    plt.show()
plot_hist(0,70,5,65)
plt.plot(c.iloc[1,:186])
plot_hist(1,50,5,45)
plt.plot(c.iloc[2,:186])
plot_hist(2,50,5,45)
plt.plot(c.iloc[3,:186])
plot_hist(3,60,15,45)
plt.plot(c.iloc[4,:186])
plot_hist(4,50,15,35)
def add_gaussian_noise(signal):
    noise = np.random.normal(0, 0.5, 186)
    return (signal + noise)
class_samples = c[c.iloc[:, 187] == 1]
tempo=c.iloc[0,:186]
bruiter=add_gaussian_noise(tempo)

plt.subplot(2,1,1)
plt.plot(c.iloc[0,:186])

plt.subplot(2,1,2)
plt.plot(bruiter)

plt.show()
target_train=train_df[187]
target_test=test_df[187]
y_train=to_categorical(target_train)
y_test=to_categorical(target_test)
X_train=train_df.iloc[:,:186].values
X_test=test_df.iloc[:,:186].values
for i in range(len(X_train)):
    X_train[i,:186]= add_gaussian_noise(X_train[i,:186])
X_train = X_train.reshape(len(X_train), X_train.shape[1],1)
X_test = X_test.reshape(len(X_test), X_test.shape[1],1)
def network(X_train,y_train,X_test,y_test):


    im_shape=(X_train.shape[1],1)
    inputs_cnn=Input(shape=(im_shape), name='inputs_cnn')
    conv1_1=Convolution1D(64, (6), activation='relu', input_shape=im_shape)(inputs_cnn)
    conv1_1=BatchNormalization()(conv1_1)
    pool1=MaxPool1D(pool_size=(3), strides=(2), padding="same")(conv1_1)
    conv2_1=Convolution1D(64, (3), activation='relu', input_shape=im_shape)(pool1)
    conv2_1=BatchNormalization()(conv2_1)
    pool2=MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv2_1)
    conv3_1=Convolution1D(64, (3), activation='relu', input_shape=im_shape)(pool2)
    conv3_1=BatchNormalization()(conv3_1)
    pool3=MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv3_1)
    flatten=Flatten()(pool3)
    dense_end1 = Dense(64, activation='relu')(flatten)
    dense_end2 = Dense(32, activation='relu')(dense_end1)
    main_output = Dense(5, activation='softmax', name='main_output')(dense_end2)


    model = Model(inputs= inputs_cnn, outputs=main_output)
    model.compile(optimizer='adam', loss='categorical_crossentropy',metrics =
['accuracy'])
```

```
    callbacks = [EarlyStopping(monitor='val_loss', patience=100, verbose=1, mode='min'),
                 ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',
save_best_only=True)]

    history=model.fit(X_train, y_train,epochs=40,callbacks=callbacks,
batch_size=32,validation_data=(X_test,y_test))
    model.load_weights('best_model.h5')
    return(model,history)
def evaluate_model(history,X_test,y_test,model):
    scores = model.evaluate((X_test),y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1]*100))

    print(history)
    fig1, ax_acc = plt.subplots()
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Model – Accuracy')
    plt.legend(['Training', 'Validation'], loc='lower right')
    plt.show()

    fig2, ax_loss = plt.subplots()
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.title('Model- Loss')
    plt.legend(['Training', 'Validation'], loc='upper right')
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.show()
    target_names=['0','1','2','3','4']

    y_true=[]
    for element in y_test:
        y_true.append(np.argmax(element))
    prediction_proba=model.predict(X_test)
    prediction=np.argmax(prediction_proba,axis=1)
    cnf_matrix = confusion_matrix(y_true, prediction)
from keras.layers import Dense, Convolution1D, MaxPool1D, Flatten, Dropout
from keras.layers import Input
from keras.models import Model
from tensorflow.keras.layers import BatchNormalization
from keras.callbacks import EarlyStopping, ModelCheckpoint

model,history=network(X_train,y_train,X_test,y_test)
evaluate_model(history, X_test, y_test, model)
y_pred = model.predict(X_test)
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

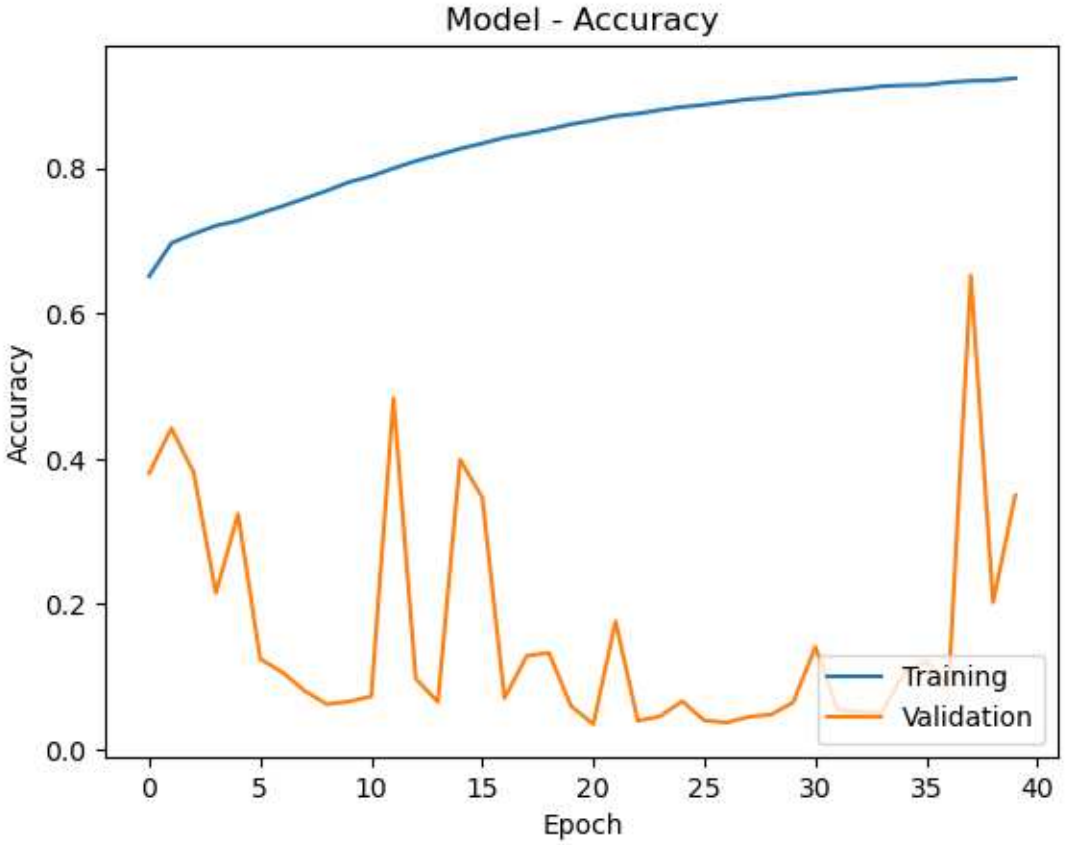
```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cnf_matrix = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))
np.set_printoptions(precision=2)

plt.figure(figsize=(10, 10))
plot_confusion_matrix(cnf_matrix, classes=['N', 'S', 'V', 'F', 'Q'],normalize=True,
                      title='Confusion matrix, with normalization')
plt.show()
import tkinter as tk
from tkinter import filedialog
import os
import numpy as np
import pandas as pd
import wfdb
from keras.models import load_model
from keras.utils.np_utils import to_categorical
import matplotlib.pyplot as plt
import tensorflow as tf
from scipy.signal import find_peaks

class ECGClassifierGUI:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Класифікатор ЕКГ сигналів")


        self.btn_select_data = tk.Button(self.root, text="Вибрати дані",
command=self.select_data)
        self.btn_classify = tk.Button(self.root, text="Класифікувати",
command=self.classify_signal)
        self.btn_find_peaks = tk.Button(self.root, text="Знайти зубці QR",
command=self.find_peaks)
        self.canvas = tk.Canvas(self.root, width=800, height=400)
        self.result_label = tk.Label(self.root, text="", font=("Helvetica", 16))

        self.info_label = tk.Label(self.root, text="", font=("Helvetica", 12),
justify="left")
        self.coordinate_label = tk.Label(self.root, text="", font=("Helvetica", 12),
justify="left")

        self.btn_select_data.grid(row=0, column=0, padx=10, pady=10)
        self.btn_classify.grid(row=1, column=0, padx=10, pady=10)
        self.btn_find_peaks.grid(row=2, column=0, padx=10, pady=10)
        self.canvas.grid(row=0, column=1, rowspan=4, padx=10, pady=10)
        self.result_label.grid(row=4, column=0, columnspan=2, padx=10, pady=10)
        self.info_label.grid(row=0, column=2, columnspan=2, padx=10, pady=10,
sticky="w")
```

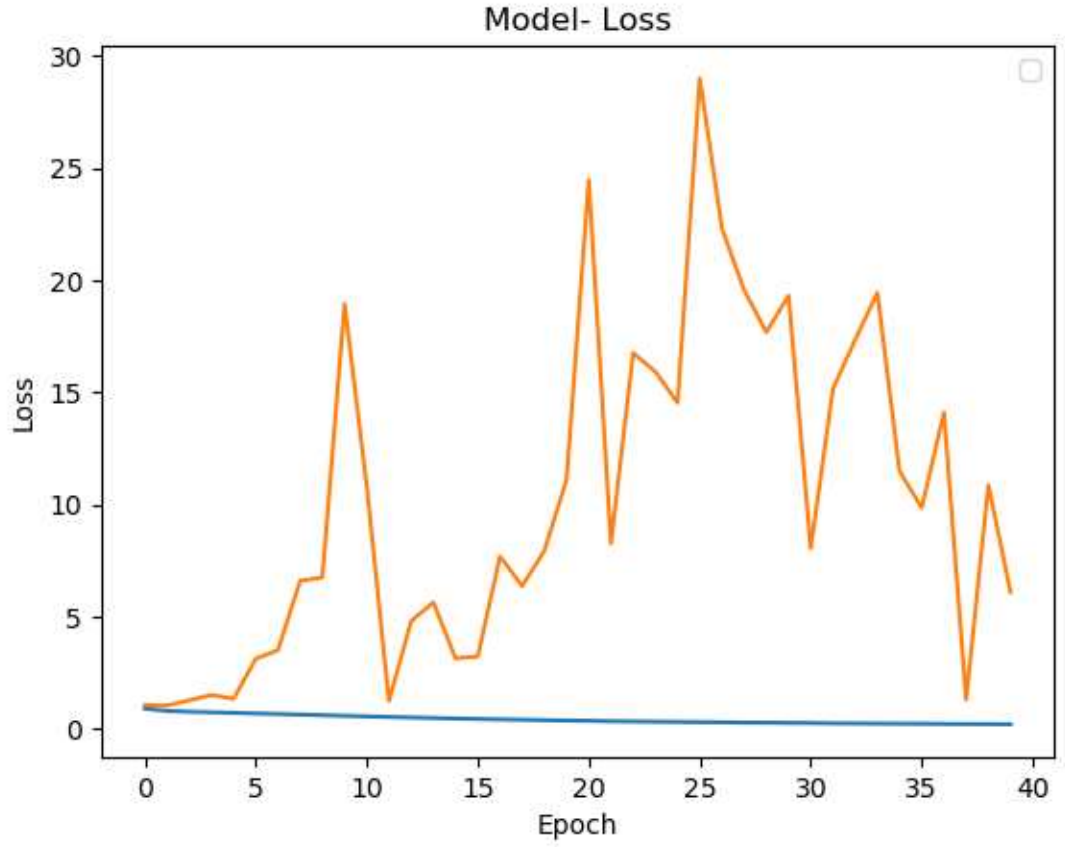
```
        self.coordinate_label.grid(row=1, column=2, columnspan=2, padx=10, pady=10,
sticky="w")
        self.model = None
        self.data = None
        self.load_model()
        self.root.mainloop()
    def load_model(self):
        model_path = 'C:/Users/Ihor/ECG/best_model.h5'
        self.model = load_model(model_path)
    def select_data(self):
        file_path = filedialog.askopenfilename(filetypes=[("Дані MIT-BIH", "*.dat")])
        if file_path:
            self.result_label.configure(text="Обрані дані: " + file_path)
            record_name = os.path.splitext(os.path.basename(file_path))[0]
            header_path = os.path.join(os.path.dirname(file_path), record_name + '.hea')
            if not os.path.isfile(header_path):
                print("Файл заголовка не знайдено для обраного запису.")
                self.show_error_message("Не вибрано жодних даних. Будь ласка, спочатку
виберіть дані.")
                return
            record_path = os.path.join(os.path.dirname(file_path), record_name)
            record = wfdb.rdrecord(record_path)
            self.data = record.p_signal[:, 0]
            print("Дані успішно завантажено.")
            self.canvas.delete("all")
            self.display_info(record)
            self.display_coordinates(record)
            self.plot_signal()
    def plot_signal(self):
        if self.data is None:
            print("Не вибрано жодних даних. Будь ласка, спочатку виберіть дані.")
            self.show_error_message("Не вибрано жодних даних. Будь ласка, спочатку
виберіть дані.")
            return
        self.canvas.delete("all")
        signal = self.data - np.mean(self.data)
        signal /= np.max(np.abs(signal))
        signal *= 180


        x = np.arange(len(signal))
        y = 200 - signal


        for i in range(len(x) - 1):
            self.canvas.create_line(x[i], y[i], x[i + 1], y[i + 1], fill="blue")

    def display_info(self, record):
        info = f"Назва запису: {record.record_name}\n" \
               f"Кількість каналів: {record.n_sig}\n" \
               f"Частота дискретизації: {record.fs} Гц"
        self.info_label.configure(text=info)

    def display_coordinates(self, record):
        coordinates = ""
        for i, sig_name in enumerate(record.sig_name):
            coordinates += f"Канал {i+1}: {sig_name}\n"
        self.coordinate_label.configure(text=coordinates)

    def classify_signal(self):
        if self.data is None:
            print("Класифікація неможлива. Не вибрано жодних даних. Будь ласка, спочатку
виберіть дані.")
```

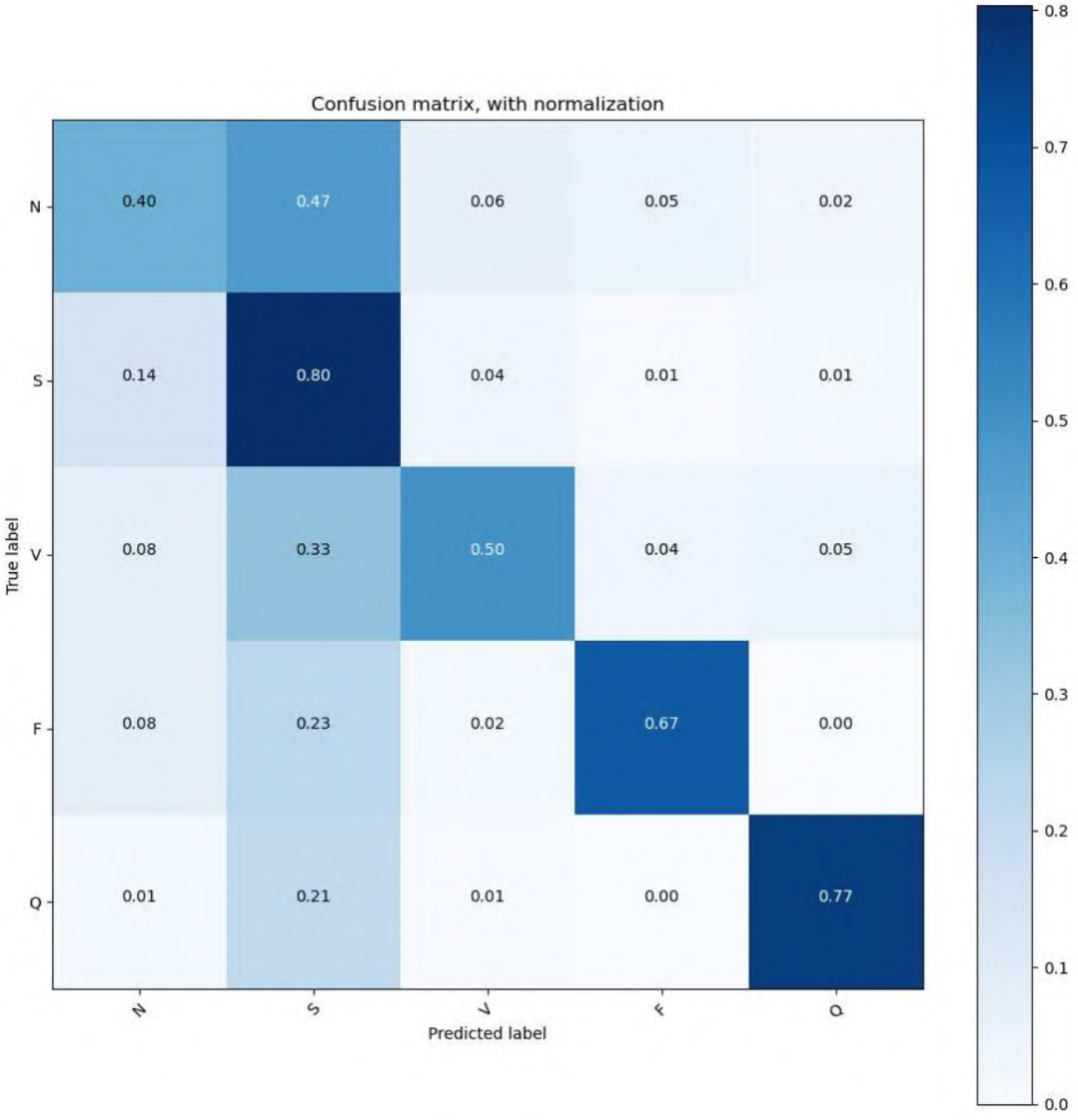
```
            self.show_error_message("Класифікація неможлива. Не вибрано жодних даних.
Будь ласка, спочатку виберіть дані.")
            return
        X = self.data[:186]
        X = X.reshape(1, len(X), 1)
        y_pred = self.model.predict(X)
        class_labels = ['N', 'S', 'V', 'F', 'Q']
        prediction = class_labels[np.argmax(y_pred)]
        self.result_label.configure(text="Клас: " + prediction)
    def find_peaks(self):
        if self.data is None:
            print("Пошук зубців неможливий. Не вибрано жодних даних. Будь ласка,
спочатку виберіть дані.")
            self.show_error_message("Пошук зубців неможливий. Не вибрано жодних даних.
Будь ласка, спочатку виберіть дані.")
            return
        signal = self.data - np.mean(self.data)
        signal /= np.max(np.abs(signal))
        signal *= 180
        r_peaks, _ = find_peaks(signal, prominence=0.5, distance=100)
        q_peaks, _ = find_peaks(-signal, prominence=0.5, distance=100)
        self.plot_signal()
        for peak in r_peaks:
            x = peak
            y = 200 - signal[peak]
            self.canvas.create_text(x, y, text="R", fill="red", font=("Helvetica", 8),
anchor="s")
        for peak in q_peaks:
            x = peak
            y = 200 - signal[peak]
            self.canvas.create_text(x, y, text="Q", fill="green", font=("Helvetica", 8),
anchor="s")
    def show_error_message(self, message):
        tk.messagebox.showerror("Помилка", message)
if __name__ == "__main__":
    ecg_classifier_gui = ECGClassifierGUI()
```

Додаток В

Результати навчання та тестування нейромережевої моделі






Confusion matrix, with normalization
N
S
V
F
Q
True label
0.40
0.47
0.06
0.05
0.02
0.14
0.80
0.04
0.01
0.01
0.08
0.33
0.50
0.04
0.05
0.08
0.23
0.02
0.67
0.00
0.01
0.21
0.01
0.00
0.77
N
S
V
F
Q
Predicted label
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

Додаток Г

Презентаційний матеріал

КВАЛІФІКАЦІЙНА робота Бакалавра

Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі

Виконав:

студент 4 курсу, групи КН-19-2

Федорчук Ігор Ігорович

Керівник:

старший викладач кафедри КН

Радюк Павло Михайлович

Актуальність

Актуальність використання автокодерів для аналізу сигналів ЕКГ полягає в тому, що вони можуть надати цінну інформацію про стан здоров'я пацієнта. Автокодери можна використовувати для виявлення аритмій, ідентифікації шаблонів у сигналі або прогнозування майбутніх значень. Автокодери також можна використовувати для виявлення аномалій у сигналі, таких як зміни частоти серцевих скорочень або аномальні ритми.

Нейронні мережі автокодерів використовують для аналізу сигналів електрокардіограми завдяки їхній здатності стискати та реконструювати вихідні дані з мінімальною втратою інформації. Автокодери можна використовувати для виділення ознак, класифікації, виявлення аномалій та інших завдань, пов'язаних із моніторингом здоров'я серця. Порівняно з традиційними методами, такими як вейвлет-перетворення або методи аналізу Фур'є, автокодери пропонують кілька переваг, таких як швидший час навчання, краща точність і більша гнучкість щодо того, які характеристики можна витягти із сигналу. Крім того, автокодувальники вимагають менше спеціальних знань порівняно з традиційними методами, що спрощує їх використання.

3

Мета і задачі роботи

Метою кваліфікаційної роботи бакалавра є покращення окреслення сигналів електрокардіограм.

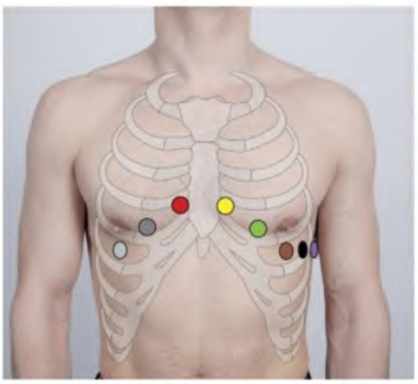
Для досягнення поставленої мети визначено наступні завдання:

- Провести аналіз методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм та обрати найкращий.
- Застосувати обраний спосіб цифрового опису та формалізації сигналів електрокардіограм до розв'язання задачі автоматизованого окреслення сигналів електрокардіограм.
- Реалізувати обраний спосіб у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм.
- Провести експериментальне тестування реалізованого модуля за еталонними наборами даних.

Досягнення мети кваліфікаційної роботи бакалавра полягає у створенні інформаційної системи, використання якої дасть змогу покращити процес окреслення сигналів електрокардіограм.

4

Електрокардіограма



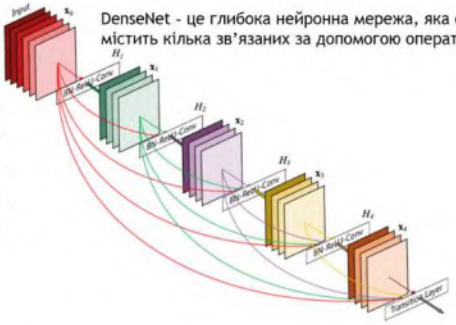
- V_1 – IV міжребер'я біля правого краю грудини
- V_2 – IV міжребер'я біля лівого краю грудини
- V_3 – між V_2 і V_4
- V_4 – V міжребер'я по лівій середньоключичній лінії
- V_5 – на рівні V_4 по передній аксилярній лінії зліва
- V_6 – на рівні V_4 по середній аксилярній лінії зліва
- V_{r3} – між V_1 та V_{r4}
- V_{r4} – V міжребер'я по правій середньоключичній лінії

Електрокардіограма – це медичний тест, який реєструє електричну активність серця. Він використовується для виявлення аномалій серця, діагностики захворювань серця та контролю ефективності лікування. ЕКГ проводиться шляхом розміщення електродів на грудях, руках і ногах

5

Проєктування архітектури автокодувальної нейромережі для окреслення сигналів електрокардіограм

Структура DenseNet базується на концепції "з'єднання" шарів, яка дозволяє передавати інформацію про кожен шар всім наступним шарам мережі. Таким чином, мережа може використовувати інформацію з усіх попередніх шарів, що поліпшує якість передбачення.



DenseNet - це глибока нейронна мережа, яка складається з блоків, кожен з яких містить кілька зв'язаних за допомогою оператора згортки.

DenseNet є однією з найефективніших архітектур глибокого навчання, яка використовується для класифікації сигналів ЕКГ на основі їх характеристик. Основна ідея DenseNet полягає в тому, щоб зберігати інформацію про всі попередні шари в кожному наступному шарі. Це досягається шляхом зв'язку кожного шару з усіма попередніми шарами в глибокій нейронній мережі

6

Функціональна структура інформаційної системи на основі автокодувальної нейромережі

Основна мета такого ПЗ - отримання з ЕКГ-сигналів максимальної кількості вимірів параметрів, а також їхнє порівняння з нормативами та іншими виміряними параметрами.

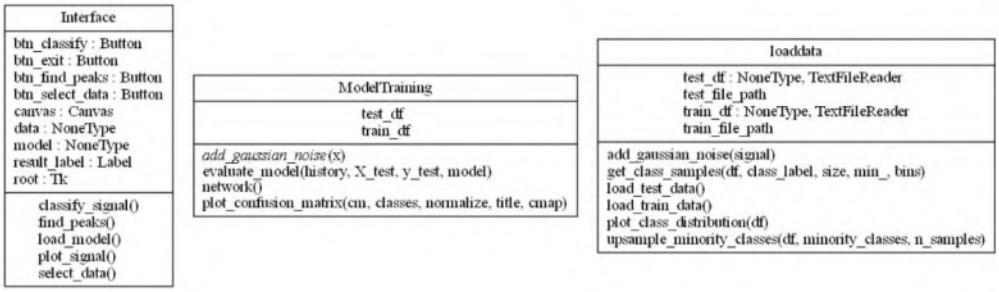
7

Набір даних

База даних про аритмію MIT-BIH містить 48 півгодинних уривків двоканальних амбулаторних записів ЕКГ, отриманих від 47 суб'єктів, які вивчалися Лабораторією аритмії BIH між 1975 і 1979 роками. Двадцять три записи були вибрані випадковим чином із набору 4000 24- години амбулаторних записів ЕКГ, зібраних у змішаної популяції стаціонарних (приблизно 60%) та амбулаторних пацієнтів (приблизно 40%) у бостонській лікарні Бет Ізраїль; решта 25 записів були відібрані з того самого набору, щоб включити менш поширені, але клінічно значущі аритмії, які не будуть добре представлені в невеликій випадковій вибірці.Записи були оцифровані зі швидкістю 360 вибірок на секунду на канал з 11-бітною роздільною здатністю в діапазоні 10 мВ. Два або більше кардіологів незалежно анотували кожен запис; розбіжності були вирішені, щоб отримати зчитувані комп'ютером довідкові анотації для кожного удару (загалом приблизно 110 000 анотацій), включені до бази даних.Цей каталог містить всю базу даних про аритмію MIT-BIH. Близько половини (25 із 48 повних записів і файли довідкових анотацій для всіх 48 записів) цієї бази даних були у вільному доступі з моменту заснування PhysioNet у вересні 1999 року. Решта 23 файли сигналів, які були доступні лише в MIT-BIH Arrhythmia База даних CD-ROM, були розміщені тут у лютому 2005 року.

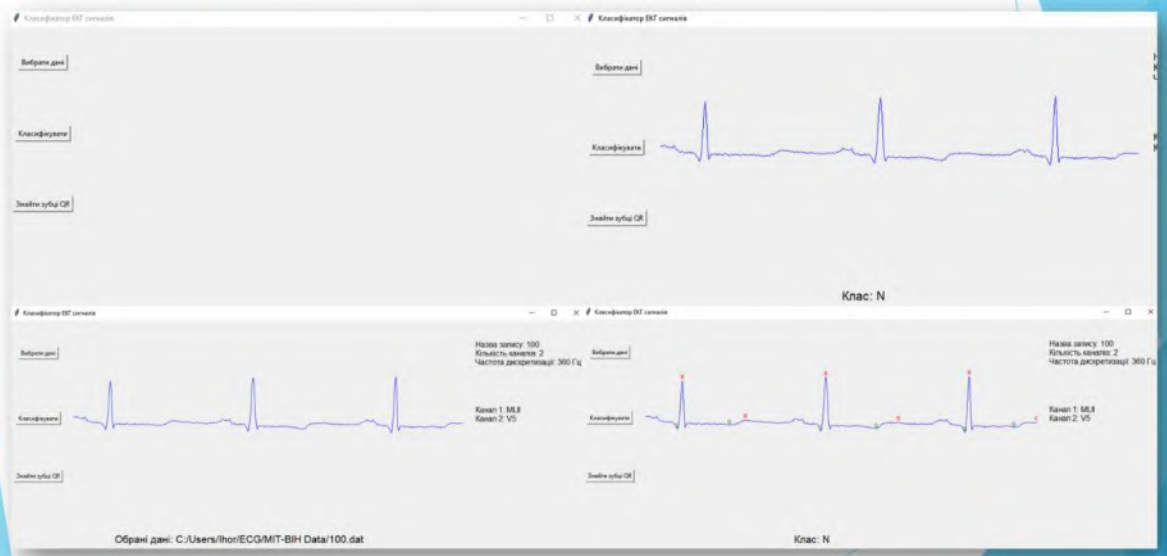
8

Діаграма класів інформаційної системи



9

Програмна реалізація



10

Висновки

В результаті виконання кваліфікаційної роботи бакалавра була досягнута поставлена мета - покращено окреслення сигналів електрокардіограм. Для досягнення цієї мети були виконані наступні завдання.

Початково був проведений детальний аналіз різних методів, способів та технологій цифрового опису й формалізації сигналів електрокардіограм з метою вибору найкращого підходу. Цей аналіз надав змогу визначити оптимальний метод опису та формалізації сигналів електрокардіограм.

Після вибору найбільш підходящого способу цифрового опису та формалізації сигналів електрокардіограм, його було успішно застосовано для розв'язання задачі автоматизованого окреслення сигналів електрокардіограм. Це дозволило досягти точного та автоматизованого окреслення сигналів електрокардіограм з високою надійністю.

Далі, обраний метод було реалізовано у вигляді модуля програмного забезпечення для автоматизованого окреслення сигналів електрокардіограм. Розроблений модуль демонструє ефективну роботу і забезпечує точне та швидке окреслення сигналів електрокардіограм.

Для підтвердження ефективності реалізованого модуля було проведено експериментальне тестування з використанням еталонних наборів даних. Результати експериментального тестування підтвердили високу точність і ефективність розробленого модуля програмного забезпечення, що підтверджує досягнення поставленої мети.

Отже, на основі проведеного аналізу, реалізації та експериментального тестування можна зробити узагальнюючий висновок, що розроблений модуль програмного забезпечення забезпечує високу точність окреслення сигналів електрокардіограм, що сприяє поліпшенню діагностики та аналізу серцево-судинних захворювань. Застосування цього модуля може допомогти медичному персоналу в ранньому виявленні та лікуванні серцевих захворювань, а також покращити загальний стан пацієнтів.

Mon May 29 18:46:44 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114256 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-05-29 Автора: І.І. Федорчук Керівники: П.М. Радюк Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	70832	1064	2005 (3%)	31 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра КН

Дата перевірки:
29.05.2023 19:42:24 EEST

Дата звіту:
29.05.2023 19:47:31 EEST

ID перевірки:
1015304290

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005671

Назва документа: **КН-19-2 Федорчук**

Кількість сторінок: 62 Кількість слів: 11535 Кількість символів: 87139 Розмір файлу: 1.63 MB ID файлу: 1014975805

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.78%
Схожість

Найбільша схожість: 1.39% з джерелом з Бібліотеки (ID файлу: 1014959353)

1.72% Джерела з Інтернету	140	Сторінка 64
2.39% Джерела з Бібліотеки	87	Сторінка 65

0.11% Цитат

Цитати	4	Сторінка 66
Посилання	1	Сторінка 66

0.01%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 0%)

0.01% Вилучення з Інтернету	1	Сторінка 67

Немає вилучених бібліотечних джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	10 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі*

Автор: *студент групи КН-19-2 Федорчук Ігор Ігорович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *док. філ., ст. викл. Радюк П.М.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Федорчука І.І., не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти програмного коду, що не мають авторства і містять поширені конструкції; поміж запозичень знаходяться загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 2.0%;

- за системою Unicheck: 2.78 %.

Отже, запозичення є допустимими та відносяться до описаних вище і адресуються до першоджерел, що, з урахуванням наведених обґрунтувань, свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ *Павло РАДЮК*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-2 Федорчука Ігоря Ігоровича*

за темою *Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі*

1. Актуальність теми

Застосування засобів штучного інтелекту, зокрема, нейронних мереж, до аналізу та окреслення сигналів електрокардіограм є перспективним напрямком досліджень у галузі медицини. Їхній потенціал полягає у здатності ефективно виявляти ознаки аритмії, ідентифікувати шаблони серцевих захворювань та прогнозувати значення потенційних захворювань. Тому розроблення інформаційної системи на основі автокодувальної нейромережі та її впровадження дасть змогу виконувати моніторинг стану здоров'я пацієнта точніше, що є актуальною задачею.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Відповідно до стандарту бакалавра вищої освіти України спеціальності 122 – Комп'ютерні науки, описом предметної галузі, об'єктом та предметом вивчення є математичні, інформаційні та імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи й технології отримання, зберігання, обробки, передачі та використання інформації. Метою поданої роботи є покращення окреслення сигналів електрокардіограм. Мету роботи досягнуто завдяки використанню методів, засобів та технологій розв'язання теоретичних і прикладних задач, що виникають у процесі проєктування та розроблення інформаційних технологій. Отже, результати виконання кваліфікаційної роботи відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

Під час виконання кваліфікаційної роботи студент Федорчук І.І. проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені перед ним завдання. Студент опанував компетентності та засвоїв результати навчання, що відповідають виконанню освітньо-професійної програми рівня вищої освіти «Бакалавр» за спеціальністю 122 – Комп'ютерні науки.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Студент особисто одержав результати кваліфікаційної роботи та обґрунтував їхню практичну значущість внаслідок виконання ним усіх поставлених завдань.

5. Ступінь оволодіння методами дослідження

У процесі аналізу способу цифрового окреслення сигналів електрокардіограм та проєктування і розроблення на його основі інформаційної системи студент Федорчук І.І. продемонстрував задовільний рівень компетентностей та володіння необхідними інструментами й обладнанням, методами, методиками та технологіями галузі інформаційних технологій.

6. Повнота та якість розкриття теми роботи

Тема роботи повністю обґрунтована й розкрита; актуальність предметної галузі та відомі дослідження щодо обраної тематики проаналізовані достатньо. Студентом успішно виконані усі поставлені перед ним завдання. Розроблене студентом програмне забезпечення для валідації та верифікації інформаційної системи цифрового окреслення сигналів електрокардіограм відповідає технічним вимогам спеціальності 122 – Комп'ютерні науки.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Матеріал кваліфікаційної роботи Федорчука І.І. подано логічно, послідовно, аргументовано та є таким, що відповідає поставленій меті. Мова і стиль викладення роботи відповідають стандартам, що забезпечує доступність сприймання матеріалу і відповідає вимогам до сучасних кваліфікаційних робіт.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблене в роботі програмне забезпечення може бути використане медичними фахівцями або їхніми асистентами під час діагностування серцево-судинних захворювань в поза клінічних умовах, наприклад, для надання додаткової інформації про стан серця в режимі реального часу в разі відсутності спеціалізованого медичного обладнання.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

З огляду на рівень виконання та забезпечення всіх необхідних вимог, вважаю, що кваліфікаційна робота Федорчука Ігоря Ігоровича може бути допущена до захисту. Рекомендована оцінка – «задовільно».

Керівник _____ *док. філ., ст. викл. каф. КН Павло РАДЮК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-2 Федорчука Ігоря Ігоровича*
за темою: *Окреслення сигналів електрокардіограм з використанням автокодувальної нейромережі*

1. Актуальність обраної теми

Автоматизоване оброблення електрокардіограм дає змогу отримати нову інформацію про стан серцево-судинної системи пацієнта, що може допомогти в точному діагностуванні та плануванні лікування. Використовуючи методи, способи та технології глибокого навчання, як от, автокодувальна нейронна мережа, можна якісніше та точніше описати стан серця, що було б неможливою з допомогою лише традиційних методів діагностування серцево-судинних захворювань.

2. Повнота розкриття мети та завдань роботи

У процесі виконання кваліфікаційної роботи були детально розкриті мета та завдання. Автором проведено глибокий аналіз процесів опису, формалізації та окреслення сигналів електрокардіограм, що здійснюється під час серцево-судинних захворювань. Проведено систематизацію основних принципів та методів, що використовуються в задачах аналізу медичних сигналів. Визначено ключові виклики та труднощі, які трапляються дослідникам та медичним фахівцям у процесі аналізу електрокардіограм. Загалом, визначені в роботі завдання є повними та збалансованими, а мета роботи є повністю розкритою.

3. Зміст кожного розділу роботи

У першому розділі кваліфікаційної роботи досліджено процес окреслення сигналів електрокардіограм засобами штучного інтелекту. Такий підхід допомагає уточнити діагноз, виявити аномалії та забезпечити швидку та точну оцінку стану серцево-судинної системи. У другому розділі використано спосіб цифрового окреслення сигналів електрокардіограм для діагностики та моніторингу серцевих захворювань. Спосіб дає можливість отримати детальну інформацію про електричну активність серця, виявити різноманітні патології та аритмії серця. Третій розділ описує процес розроблення та реалізації комплексної системи, яка забезпечує завантаження, обробку, класифікацію та візуалізацію ЕКГ-сигналів для виявлення аномалій та аналізу ритму серця.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений модуль програмного забезпечення забезпечує високу точність окреслення сигналів електрокардіограм, що сприяє поліпшенню діагностики та аналізу серцево-судинних захворювань. Розроблений модуль має потенціал для подальшого розширення та вдосконалення, наприклад, через додання різних особистісних показників пацієнта.

5. Якість оформлення кваліфікаційної роботи бакалавра

Подана кваліфікаційна робота має задовільний рівень якості оформлення і висвітлення матеріалу. Автор вдало засвоїв вимоги до структури, стилю та оформлення науково-дослідницької роботи. Як наслідок, заголовки розділів є чіткими та зрозумілими, а структура роботи є логічно впорядкованою.

6. Недоліки кваліфікаційної роботи бакалавра

Робота також містить недоліки. Записка містить орфографічні та граматичні помилки, які негативно впливають на читабельність. Автор надав недостатньо посилань на авторитетні джерела, що підтримували б його аргументи. Взаємодію користувача із розробленою інформаційною системою описано недостатньо.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

*З огляду на рівень виконання та забезпечення всіх необхідних вимог, вважаю, що подана кваліфікаційна робота бакалавра може бути допущена до захисту. Рекомендована оцінка – **«задовільно»**.*

Рецензент к.ф.-м.н., доц. каф. ВМКЗ Рамський А.О.