

Вінницький національний технічний університет

Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

МИЛОСЕРДОВ ДМИТРО АНДРІЙОВИЧ

УДК 004.8:004.032.26

ДИСЕРТАЦІЯ
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ НА ОСНОВІ СПАЙКІНГОВИХ
НЕЙРОННИХ МЕРЕЖ ДЛЯ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ

Спеціальність 122 «Комп'ютерні науки»

Галузь знань 12 «Інформаційні технології»

Подається на здобуття ступеня доктора філософії

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

_____Д.А. Милосердов

Науковий керівник:

Колесницький Олег Костянтинович
кандидат технічних наук, професор

Вінниця – 2026

АНОТАЦІЯ

Милосердов Д. А. Інформаційна технологія на основі спайкінгових нейронних мереж для класифікації сигналів ЕКГ. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії в галузі знань 12 Інформаційні технології за спеціальністю 122 «Комп'ютерні науки» – Вінницький національний технічний університет, Вінниця, 2026.

В дисертаційній роботі розв'язана наукова задача розробки інформаційної технології, методів та програмних засобів класифікації сигналів ЕКГ з використанням спайкінгових нейронних мереж, що дозволило підвищити точність систем класифікації сигналів ЕКГ, які також є ресурсоефективними та мають реальночасову швидкодію.

Проблема точної класифікації ЕКГ-сигналів є актуальною через зростання кількості пацієнтів з серцевими захворюваннями та необхідність у портативних пристроях для моніторингу в реальному часі. Традиційні методи аналізу ЕКГ, такі як візуальна інтерпретація лікарями, є суб'єктивними та трудомісткими, що призводить до помилок у 20-30% випадків, особливо при виявленні рідкісних аритмій. Автоматизовані системи на основі штучного інтелекту дозволяють підвищити точність до 95% і більше, зменшуючи навантаження на медичний персонал.

У контексті низько-енергетичних пристроїв, таких як носимі гаджети (наприклад, смарт-годинники), класифікація ЕКГ повинна бути енергоефективною та здатною до обробки даних у реальному часі. Відомо, що традиційні нейронні мережі такі, як згорткові чи рекурентні, споживають значну обчислювальну потужність, що робить їх непридатними для вбудованих систем з обмеженими ресурсами. Спайкінгові нейронні мережі найбільше відповідають біологічним процесам у мозку, пропонують альтернативу, забезпечуючи низьке енергоспоживання завдяки асинхронній обробці спайків.

В дисертаційній роботі описано узагальнену схему класифікації сигналів ЕКГ, проаналізовано інтерпацієнтську (inter-patient) варіабельність сигналів ЕКГ. Здійснено класифікацію відомих методів та засобів класифікації сигналів ЕКГ та сформульовано перспективи їх вдосконалення, що полягають у використанні класифікаторів на основі спайкінгових нейронних мереж. Здійснено аналіз відомих баз даних сигналів ЕКГ та показників точності, що використовуються при оцінюванні якості систем класифікації сигналів ЕКГ. Виділено недоліки існуючих систем класифікації сигналів ЕКГ, запропоновано напрямки їх вдосконалення та сформульовано задачі досліджень.

Обґрунтовано перспективність використання спайкінгових нейронних мереж для класифікації сигналів ЕКГ, оскільки вони ресурсоефективні, працюють у реальному часі та підходять для носимих пристроїв. Проаналізовано моделі спайкінгових нейронів з точки зору використання їх у процесі класифікації сигналів ЕКГ. Для використання обрана модель LIF-нейрона з входами збуджувальних та гальмівних синапсів, що дозволяє використовувати імпульси однакової полярності для передачі як збуджувальних, так і гальмівних сигналів.

Розроблено структуру та архітектуру спайкінгової нейронної мережі включно з принципами її побудови та функціонування з орієнтацією на застосування у процесі класифікації сигналів ЕКГ. Особливістю розробленої архітектури є те, що вона має «резервуарне обчислення» та оптимізовані параметри з'єднання нейронів.

Розроблено метод навчання спайкінгової нейронної мережі та метод попередньої обробки та кодування вхідної інформації в процесі класифікації сигналів ЕКГ. Особливістю попередньої обробки та кодування є процес нормалізації сигналів ЕКГ, який здійснюється як по амплітуді, так і по часу, а для кодування у спайки використовуються Гаусові рецептивні поля.

Особливістю процесу навчання є те, що навчання має 3 етапи. Першим етапом є початкове закріплення вихідних нейронів за класами серцебиття, що

сприяє загальному пришвидшенню процесу навчання. Другий етап є звичайним навчанням з учителем. А останній етап дозволяє більш тонко налаштувати ваги для підвищення точності класифікації. Також розроблено загальний метод класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.

Розроблено програмне забезпечення для моделювання та оцінювання точності запропонованого методу класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі. Описано програмну реалізацію методу класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі на мові Python у середовищі розробки Visual Studio Code з використанням бібліотек PyTorch та SpikingJelly. Розглянуто методику оцінювання точності методів класифікації сигналів ЕКГ, що складається з таких етапів: вибір референсної бази даних (було обрано БД МІТ-ВІН), аналіз показників якості систем класифікації сигналів ЕКГ, розробка протоколу проведення експериментальних досліджень. Проведено експериментальні дослідження точності методу класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.

Тестування на сигналах ЕКГ від різних пацієнтів показало, що розроблена інформаційна технологія має загальну точність класифікації 96,48%, чутливість (macro recall) 96,48%. Загалом, у відносних показниках, запропонована система за точністю краща за референсну на 1.39%, за чутливістю залежно від класу на 0.05-10% та за влучністю на 0.21-25.73% при тестуванні на сигналах ЕКГ різних пацієнтів. Визначено перспективи подальших досліджень, які полягають як у перенесенні моделі на нейроморфне апаратне забезпечення, оптимізації моделі для носимих пристроїв, онлайн-навчанні вихідного зчитувального шару на потокових даних, розширенням на інші бази даних та задачі інтеграції з реальним медичним обладнанням.

Ключові слова: інформаційна технологія, спайкінгові нейронні мережі, класифікація сигналів ЕКГ, електрокардіограма, нейромережевий класифікатор, штучний інтелект, машинне навчання, глибоке навчання, аналіз сигналів, нейромережева технологія.

ABSTRACT

Myloserdov D. A. Information Technology Based on Spiking Neural Networks for ECG Signal Classification. – Qualification scientific work on manuscript rights.

Thesis for the degree of Doctor of Philosophy in the field of knowledge 12 Information Technologies, specialty 122 “Computer Science” – Vinnytsia National Technical University, Vinnytsia, 2026.

The dissertation solves the scientific problem of developing an information technology, methods, and software tools for ECG signal classification using spiking neural networks. This approach improves the accuracy of ECG signal classification systems while ensuring high resource efficiency and real-time performance.

The problem of accurate ECG signal classification remains highly relevant due to the increasing number of patients with cardiovascular diseases and the growing need for portable devices capable of real-time monitoring. Traditional methods of ECG analysis, such as visual interpretation by physicians, are subjective and time-consuming, leading to diagnostic errors in 20-30% of cases, particularly in detecting rare arrhythmias. Automated artificial intelligence-based systems can increase accuracy to 95% and above, significantly reducing the workload on medical personnel.

In the context of low-power devices such as wearable gadgets (e.g., smartwatches), ECG classification must be energy-efficient and capable of processing data in real time. Conventional neural networks, such as convolutional or recurrent networks, consume substantial computational resources, making them unsuitable for embedded systems with limited power and hardware capabilities. Spiking neural networks, which most closely emulate biological processes in the brain, offer a promising alternative by providing low energy consumption through asynchronous spike-based processing.

The dissertation presents a generalized scheme for ECG signal classification and analyzes inter-patient variability of ECG signals. A systematic classification of

existing methods and tools for ECG signal classification is performed, and prospects for their improvement through the use of spiking neural network classifiers are formulated. The study includes an analysis of known ECG signal databases and performance metrics used to evaluate classification systems. Shortcomings of existing ECG classification systems are identified, directions for their improvement are proposed, and research objectives are defined.

The feasibility of using spiking neural networks for ECG signal classification is substantiated, as they are resource-efficient, operate in real time, and are well-suited for wearable devices. Spiking neuron models are analyzed with respect to their applicability in ECG signal classification. The Leaky Integrate-and-Fire (LIF) neuron model with excitatory and inhibitory synapses was selected, enabling the use of uniform-polarity pulses to transmit both excitatory and inhibitory signals.

A structure and architecture of a spiking neural network designed for ECG signal classification is developed, including principles of construction and operation. A distinctive feature of the proposed architecture is the integration of reservoir computing principles and optimized neuron connection parameters.

A training method for the spiking neural network and a method for preprocessing and encoding input information for ECG signal classification are developed. The preprocessing includes normalization of ECG signals by both amplitude and time, while Gaussian receptive fields are used for spike encoding.

The training process consists of three stages. The first stage involves initial assignment of output neurons to heartbeat classes, which accelerates overall training. The second stage is standard supervised learning. The final stage enables fine-tuning of synaptic weights to improve classification accuracy. A general method for ECG signal classification based on the spiking neural network is also proposed.

Software for modeling and evaluating the accuracy of the proposed ECG signal classification method based on spiking neural networks has been developed. The software implementation was realized in Python using the Visual Studio Code environment with the PyTorch and SpikingJelly libraries. A methodology for assessing the reliability of ECG classification methods is described, which includes

selection of a reference database (MIT-BIH database), analysis of quality metrics, and development of an experimental research protocol. Experimental validation of the proposed spiking neural network-based ECG classification method was conducted.

Testing on ECG signals from different patients demonstrated that the developed information technology achieves a classification accuracy of 96.48% and a macro recall (sensitivity) of 96.48%. Overall, the proposed system outperforms the reference baseline by 1.39% in accuracy, by 0.05-10% in per-class sensitivity and by 0.21-25.73% in per-class precision when tested on inter-patient ECG signals. Prospects for further research include porting the model to neuromorphic hardware, optimizing it for wearable devices, implementing online learning of the readout layer on streaming data, expanding to other databases, and integrating the system with real medical equipment.

Keywords: information technology, spiking neural networks, ECG signal classification, electrocardiogram, neural network classifier, artificial intelligence, machine learning, deep learning, signal analysis, neural network technology.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

[1] Милосердов Д. А., Колесницький О. К., Волосович О. В., Жумагулова Ш., Короленко О. В. Класифікація електрокардіограм як динамічного процесу на основі спайкінгової нейронної мережі. Оптико-електронні інформаційно-енергетичні технології. 2024. Вип. 48, № 2. С. 68–77. DOI: 10.31649/1681-7893-2024-48-2-68-77.

[2] Милосердов Д. А., Колесницький О. К. Застосування спайкінгових нейронних мереж до класифікації електрокардіограм. Огляд. Наука і техніка сьогодні. Серія «Фізико-математичні науки». 2025. № 9(50). С. 1612–1643. DOI: 10.52058/2786-6025-2025-9(50)-1612-1643.

[3] Милосердов Д. А., Колесницький О. К. Метод попередньої обробки сигналу ЕКГ для класифікації електрокардіограм за допомогою спайкінгових нейронних мереж. Технічні науки та технології. 2025. № 3(41). С. 203–210. DOI: 10.25140/2411-5363-2025-3(41)-203-210. URL: <http://tst.stu.cn.ua/article/view/345202/332545>.

[4] Милосердов Д. А., Колесницький О. К. Метод класифікації електрокардіограм із застосуванням спайкінгових нейронних мереж. Вісник Вінницького політехнічного інституту. 2025. № 5. С. 121–128. DOI: 10.31649/1997-9266-2025-182-5-121-128

[5] Милосердов Д. А., Колесницький О. К. Інформаційна технологія класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі. Наука і техніка сьогодні. Серія «Фізико-математичні науки». 2026. № 2(56). С. 2535–2552. DOI: 10.52058/2786-6025-2026-2(56)-2535-2552.

[6] Семчук В. О., Милосердов Д. А., Колесницький О. К. Інформаційна технологія розпізнавання вимовлених літер на основі спайкінгової нейронної мережі. Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)». Вінниця, 2023. [Електронний ресурс]. URL: <https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/16854/1405> 5.

[7] Милосердов Д. А., Колесницький О. К. Розпізнавання серцевої недостатності як багатопараметричного процесу на основі спайкінгових нейронних мереж. Матеріали 1-ої Міжнародної науково-практичної конференції «Інформаційні системи та технології: результати і перспективи (IST 2024)», 6 березня 2024 р. Київ : ФІТ КНУТШ, 2024. [Електронний ресурс]. URL:

https://ist.fit.knu.ua/_files/ugd/016074_8fc56ecbbfa94cde874eedd872f3780e.pdf.

[8] Милосердов Д. А., Колесницький О. К. Метод попередньої обробки сигналу електрокардіограм для класифікації за допомогою спайкінгових нейронних мереж. Science, Technology and Industry in the Digital Age : Collection of Scientific Papers with Proceedings of the 1st International Scientific and Practical Conference. International Scientific Unity. September 24–26, 2025. Hamburg, Germany. P. 159–164. DOI: 10.70286/isu-24.09.2025.

[9] Милосердов Д. А., Колесницький О. К. Інформаційна технологія класифікації сигналів ЕКГ за допомогою спайкінгових нейронних мереж. Innovative Research in Science and Economy : Collection of Scientific Papers with Proceedings of the 4th International Scientific and Practical Conference. International Scientific Unity. Brussels, Belgium. April 1–3, 2026. P. 178–181. DOI: 10.70286/ISU-01.04.2026.009.

[10] Свідоцтво про реєстрацію авторського права на твір № 139885. Комп'ютерна програма «Інформаційна технологія класифікації часових рядів». Колесницький О. К., Денисов І. К., Милосердов Д. А. Дата реєстрації: 06.10.2025.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	12
ВСТУП	14
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ	22
1.1 Актуальність дослідження та сфери застосування інформаційних технологій ресурсоефективної реальночасової класифікації сигналів ЕКГ	22
1.2 Аналіз існуючих методів та інформаційних технологій класифікації сигналів ЕКГ	26
1.3 Аналіз існуючих методів та інформаційних технологій класифікації сигналів ЕКГ на основі спайкінгових нейронних мереж	31
1.4 Недоліки відомих методів та інформаційних технологій класифікації сигналів ЕКГ та постановка задач дослідження	51
1.5 Висновки до першого розділу	52
РОЗДІЛ 2 МОДЕЛІ ТА МЕТОДИ ДЛЯ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ НА ОСНОВІ СПАЙКІНГОВИХ НЕЙРОННИХ МЕРЕЖ	54
2.1 Моделі спайкінгових нейронів, що використовуються при класифікації сигналів ЕКГ	54
2.2 Архітектура спайкінгової нейронної мережі	59
2.3 Розробка методу навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ	63
2.4 Розробка методу попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу	76
2.5 Розробка методу класифікації сигналу ЕКГ на основі спайкінгової нейронної мережі	83
2.6 Висновки до другого розділу	85
РОЗДІЛ 3 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ НА ОСНОВІ СПАЙКІНГОВИХ НЕЙРОННИХ МЕРЕЖ ..	87
3.1 Загальна схема та етапи інформаційної технології	87

	11
3.2 Обґрунтування вибору набору даних для навчання та тестування спайкінгової нейронної мережі	89
3.3 Дослідження варіабельності форми сигналів ЕКГ різних класів аритмії	91
3.4 Вибір бібліотек та програмного середовища для реалізації інформаційної технології класифікації сигналів ЕКГ	98
3.5 Програмна реалізація інформаційної технології для класифікації сигналів ЕКГ	109
3.6 Висновки до третього розділу	118
РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ НА ОСНОВІ СПАЙКІНГОВОЇ НЕЙРОННОЇ МЕРЕЖІ	120
4.1 Параметри тестування інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі	120
4.2 Методика оцінювання та показники точності класифікації сигналів ЕКГ	123
4.3 Результати експериментальних досліджень інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі	128
4.4 Оцінювання ресурсоефективності та реально-часової швидкодії розробленої інформаційної технології	139
4.5 Перспективи подальших досліджень	141
4.6 Висновки до четвертого розділу	143
ЗАГАЛЬНІ ВИСНОВКИ	145
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	148
ДОДАТКИ	163
Додаток А Акти впровадження результатів дисертації	164
Додаток Б Фрагмент лістингу програми	168
Додаток В Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації	183

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AAMI – Association for the Advancement of Medical Instrumentation

ANN – Artificial Neural Networks

ASIC – Application-Specific Integrated Circuit

AUC – Area Under Curve

CNN – Convolutional Neural Networks

DNN – Deep Neural Networks

DT – Decision Trees

ECG – Electrocardiogram

ЕКГ – електрокардіограма

EEG – електроенцефалографія

F1-score – гармонічне середнє влучності та чутливості

FPGA – Field-Programmable Gate Array

FT – Fourier Transform

G-STBP – Gradient-based Surrogate Training for Backpropagation

ICA – Independent Component Analysis

ILR – Implantable Loop Recorder

IoT – Internet of Things

LDA – Linear Discriminant Analysis

LIF – Leaky Integrate-and-Fire

Macro-Recall (macro recall) – середня чутливість по класах

MIT-BIH – MIT-BIH Arrhythmia Database

MPS – Metal Performance Shaders

N_LSM – кількість нейронів у резервуарі

N_INPUT, N_OUTPUT – кількість вхідних/вихідних нейронів

PCA – Principal Component Analysis

PR-AUC – Precision-Recall Area Under Curve

R-STDP – Reward-Modulated Spike-Timing Dependent Plasticity

RNN – Recurrent Neural Networks

ROC – Receiver Operating Characteristic

SNN – Spiking Neural Networks

SOP / NSOP / SynOps – Synaptic Operations

SMOTE – Synthetic Minority Over-sampling Technique

SSF – Sum-Spikes-Fire

STBP – Spatio-Temporal BackPropagation

STDP – Spike-Timing Dependent Plasticity

SVM – Support Vector Machine

t-SNE – t-distributed Stochastic Neighbor Embedding

VS Code – Visual Studio Code

WT – Wavelet Transform

DECAY_INPUT, DECAY_LSM – коефіцієнти затухання

DT – часовий крок симуляції

SPARSITY – щільність зв'язків у резервуарі

TAU_LSM – часова константа мембрани резервуару

TP, TN, FP, FN – True Positive / True Negative / False Positive / False Negative

ВОЗ – Всесвітня організація охорони здоров'я

ССЗ – серцево-судинні захворювання

FF – Flip-Flop

LUT – Look-up Table

ВСТУП

Обґрунтування вибору теми дослідження. Біомедичні інформаційні технології – це галузь, яка застосовує інженерні принципи для розв’язання медичних проблем і покращення охорони здоров’я. Галузь біомедичної інженерії включає автоматичну діагностику мозку, нервової системи, серця та скелетної системи, клінічне лікування, а також розробку ліків і пристроїв для обстеження та лікування. Автоматична діагностика з використанням біосигналів у поєднанні з носимими пристроями привертає увагу як інноваційна галузь, яка може захищати здоров’я багатьох людей шляхом постійного моніторингу стану здоров’я особи та раннього виявлення потенційно небезпечних ситуацій. Галузь автоматичної діагностики включає діагностику захворювань черепно-мозкових нервів за допомогою електроенцефалографії (ЕЕГ), серцево-судинних захворювань за допомогою електрокардіограми (ЕКГ) та первинних станів здоров’я за допомогою артеріального тиску та насичення киснем. Особливо важливою є технологія автоматичної діагностики аритмій з використанням сигналів ЕКГ, що досліджується багатьма вченими, оскільки вона відіграє ключову роль у ранньому виявленні аритмії, яка може призвести до загрозливих для життя серцево-судинних захворювань, у повсякденному житті без відвідування лікарні.

Аритмія – стан, при якому серце має нерегулярний ритм або аномальну частоту серцевих скорочень. Існують різні види аритмій, включаючи шлуночкові ектопічні удари та надшлуночкові ектопічні удари. Хоча не всі типи аритмій є загрозливими для життя, важливо рано виявляти аритмії, які є провісниками серцево-судинних захворювань.

У традиційній системі діагностування аритмій пацієнт, який виявляє аномальні симптоми, відвідує медичний центр і вимірює ЕКГ протягом короткого періоду або носить прикріплені електроди і холтер протягом 24 годин і вимірює ЕКГ, а кардіолог аналізує виміряну протягом тривалого

періоду ЕКГ. Ця традиційна система має обмеження в тому, що є незручною і потребує багато часу для пацієнтів, а медичний центр також потребує багатьох ресурсів для аналізу. Крім того, оскільки аритмії виникають раптово та переривчасто, важко ефективно виявляти аномальні серцеві удари за допомогою поточної системи. Як зусилля для того, щоб зробити можливим раннє виявлення аритмії в повсякденному житті, було проведено багато досліджень для постійного моніторингу сигналів ЕКГ та їх автоматичної класифікації.

Багато вітчизняних та закордонних вчених досліджували наукову задачу класифікації сигналів ЕКГ. В Україні відомі такі наукові школи як Інститут кібернетики НАН України (Прімін Михайло Андрійович, Будник Микола Миколайович, Чайковський Ілля Анатолійович), Київський національний університет імені Тараса Шевченка (Панченко Тарас Володимирович), Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського" (Іванушкіна Наталія Георгіївна), Вінницький Національний Технічний Університет (Павлов Сергій Володимирович), Хмельницький національний університет (Бармак Олександр Володимирович) та інші. Закордоном є дуже багато наукових шкіл, але оскільки я досліджував напрямок класифікації ЕКГ на основі спайкінгових нейронних мереж, то можна назвати таких вчених як Kyung Ki Kim (Daegu University), Zhuo Zou, Lirong Zheng (Fudan University), Sounak Dey (TCS Research), Matin Hashemi (Sharif University of Technology), Yan Chiew Wong (Universiti Teknikal Malaysia Melaka), Weng-Fai Wong (National University of Singapore) та інші.

Методи автоматичної класифікації аритмій поєднуються з функцією вимірювання ЕКГ носимих пристроїв, які останнім часом широко використовуються, уможливорюючи моніторинг аритмії без допомоги спеціаліста, при цьому зручно та портативно в повсякденному житті. Вимоги до автоматизованої класифікації аритмій – функціонування в реальному часі, низьке споживання енергії та низька обчислювальна складність, що необхідно

для роботи в носимих пристроях, яким абсолютно бракує ємності джерела живлення та обчислювальної потужності.

Однак класифікація сигналів ЕКГ має багато невирішених проблем та викликів. Однією з найбільших проблем є той факт, що більшість досліджень фокусується на точності, а не на чутливості класифікації. Саме ця характеристика має більше значення в медицині, оскільки чутливість відповідає за зменшення невиявлених випадків патології. Також багато робіт мають високу точність на сигналах одного пацієнта (inpatient accuracy), а інтерпацієнтські (interpatient) показники точності не досягають таких самих високих значень (>98%). Саме вирішенню цих проблем і присвячено дане дослідження, тому воно є вельми актуальним.

Зв'язок роботи з науковими програмами, планами, темами

Дисертаційне дослідження проводилось згідно з планами науково-дослідних робіт кафедри комп'ютерних наук Вінницького національного технічного університету, в тому числі в межах:

1. Кафедральної НДР 22 К2 «Моделювання складних систем та розробка прикладних інформаційних технологій» (2025-2029);
2. Кафедральної НДР 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» (2022-2024).

Мета і завдання дослідження. Метою дисертаційного дослідження є підвищення точності класифікації ЕКГ шляхом розробки вдосконаленої інформаційної технології класифікації сигналів ЕКГ з використанням спайкінгових нейронних мереж.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Здійснити класифікацію та порівняльний аналіз відомих методів та інформаційних технологій класифікації сигналів ЕКГ.
2. Вдосконалити архітектуру спайкінгової нейронної мережі
3. Розробити метод навчання спайкінгової нейронної мережі для використання у задачі класифікації сигналів ЕКГ.

4. Розробити удосконалений метод попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу та розробити метод класифікації сигналів ЕКГ з використанням спайкінгової нейронної мережі.

5. На основі запропонованих методів розробити вдосконалену інформаційну технологію класифікації сигналів ЕКГ з використанням спайкінгової нейронної мережі.

6. Обґрунтувати вибір набору даних сигналів ЕКГ для навчання та тестування спайкінгової нейронної мережі та дослідити варіабельність сигналів ЕКГ різних класів аритмії з обраного набору даних.

7. Обґрунтувати вибір спеціалізованих бібліотек, програмного середовища та здійснити програмну реалізацію інформаційної технології для класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.

8. Здійснити тестування інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі та дослідити і кількісно оцінити показники точності класифікації сигналів ЕКГ розробленої інформаційної технології, використовуючи дані тестування, та порівняти їх з показниками кращих із відомих інформаційних технологій класифікації сигналів ЕКГ.

Об’єкт дослідження – процес класифікації сигналів ЕКГ з використанням спайкінгових нейронних мереж.

Предмет дослідження – методи, моделі, інформаційні технології та програмні засоби класифікації сигналів ЕКГ з використанням спайкінгової нейронної мережі, для підвищення точності процесу класифікації сигналів ЕКГ при збереженні ресурсоефективності та реально-часової швидкодії.

Методи дослідження.

При розв’язанні поставлених задач використовувались методи: системного аналізу для формулювання вимог до параметрів процесу класифікації сигналів ЕКГ; методи теорії інформації та математичної статистики для оцінювання варіабельності сигналів ЕКГ; методи математичного моделювання для удосконалення математичних моделей

спайкінгових нейронів; теорія штучних нейронних мереж для синтезу структури та розробки режимів роботи спайкінгової нейронної мережі; теорія планування експерименту для обробки даних експериментальних досліджень програмних засобів класифікації сигналів ЕКГ; теорія ймовірностей і математична статистика для оцінювання показників точності класифікації сигналів ЕКГ.

Наукова новизна отриманих результатів.

Нові наукові результати, отримані в дисертаційній роботі, такі:

1. Вперше запропоновано метод навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ, який перед основним етапом навчання з учителем містить етап кластерної ініціалізації ваг вихідних нейронів, а після основного етапу - етап тонкого налаштування ваг нейронів вихідного шару, що забезпечило покращення якості навчання спайкінгової нейронної мережі та підвищення точності класифікації сигналів ЕКГ.

2. Удосконалено метод попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу, який відрізняється від існуючих введенням етапу часової нормалізації відліків сигналу ЕКГ у поєднанні з кодуванням у спайки на основі Гаусових рецептивних полів, що дозволило зберегти більше корисної інформації при перетворенні відліків ЕКГ у спайки і тим самим підвищити точність класифікації сигналів ЕКГ, а також дозволило зменшити середню кількість спайків на інференс і тим самим сприяти економії енергії споживання.

3. Удосконалено архітектуру спайкінгової нейронної мережі для використання у задачі класифікації сигналів ЕКГ, що відрізняється від існуючих формуванням структури нейронної мережі, вибором параметрів з'єднання нейронів між шарами та у резервуарі, що забезпечило підвищення точності класифікації сигналів ЕКГ.

4. Удосконалено інформаційну технологію класифікації сигналів ЕКГ, що відрізняється від існуючих використанням штучної спайкінгової нейронної мережі удосконаленої архітектури та спеціально розроблених методів

навчання спайкінгової нейронної мережі, попередньої обробки та кодування сигналів ЕКГ, що забезпечило підвищення точності класифікації сигналів ЕКГ при збереженні ресурсоефективності та реально-часової швидкодії.

Новизна викладених у роботі наукових результатів підтверджується статтями у фахових виданнях категорії Б, затверджених МОН України [1], [2], [3], [4], [5].

Практичне значення отриманих результатів.

Практична цінність роботи полягає в тому, що запропоновано інформаційну технологію та алгоритм класифікації сигналів ЕКГ, розроблено програмні засоби та проведено тестування і експериментальні дослідження програми класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.

Теоретичні та практичні дослідження, викладені в дисертації, дозволили:

- 1) розробити структурну схему спайкінгової нейронної мережі для використання у задачі класифікації сигналів ЕКГ;
- 2) розробити алгоритм роботи та діаграму класів програми класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі;
- 3) розробити програмні засоби класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.
- 4) отримати результати тестування програми класифікації сигналів ЕКГ;
- 5) визначити числові значення показників точності класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі та порівняти їх із кращими відомими засобами класифікації сигналів ЕКГ.

Отримані результати дослідження було впроваджено в ТОВ «ВІНСТАРС ТЕХНОЛОДЖИ», ПМП «Центр-В» та в навчальний процес Вінницького національного технічного університету. Відповідні акти впровадження наведено в додатку А.

Окремі теоретичні результати дисертаційної роботи використовуються в освітньому процесі на кафедрі КН (комп'ютерних наук) ВНТУ в рамках спеціальності 122 «Комп'ютерні науки» при викладанні таких дисциплін:

«Сучасні парадигми штучних нейронних мереж» та «Нейрокомп'ютерні технології штучного інтелекту» підготовки здобувачів ступеня бакалавра, дисципліни «Нейромережеві методи штучного інтелекту» підготовки здобувачів ступеня магістра та дисципліни «Нейромережеві моделі та технології обчислювального інтелекту» підготовки здобувачів ступеня доктора філософії.

Особистий внесок здобувача.

Усі наукові положення та результати дисертаційної роботи, що виносяться на захист, отримані здобувачем самостійно. Особистий внесок здобувача в роботах, опублікованих в співавторстві, такий: в роботі [1] – запропоновано метод класифікації сигналів ЕКГ з використанням спайкінгової нейронної мережі; в роботі [2] – проаналізовано сучасні рішення для класифікації сигналів ЕКГ за допомогою спайкінгових нейронних мереж, здійснено класифікацію методів розпізнавання сигналів ЕКГ; в роботі [3] – удосконалено метод попередньої обробки сигналу ЕКГ для подальшого його використання у спайкінгових нейронних мережах, що призначені для виконання задачі класифікації електрокардіограм, протестовано запропонований метод обробки сигналу ЕКГ; в роботі [4] – проаналізовано біологічні та математичні основи спайкінгових нейронних мереж, моделі нейронів та синапсів, розглянуто основні характеристики ЕКГ та її головні ознаки, удосконалено архітектуру спайкінгової нейронної мережі, описано принципи побудови зв'язків між нейронами, що передбачає перехід від традиційних “гальмівних/збуджувальних нейронів” до “гальмівних/збуджувальних синапсів”; в роботі [5] – удосконалено інформаційну технологію для класифікації сигналів ЕКГ на основі спайкінгових нейронних мереж, запропоновано новий метод навчання спайкінгової нейронної мережі, описано результати експериментальних досліджень, порівняно розроблену систему з аналогом; в роботі [6] – обґрунтовано вибір спайкінгових нейронних мереж для розпізнавання динамічних сигналів; в роботі [7] – удосконалено метод динамічного

розпізнавання сигналів ЕКГ; в роботі [8] – запропоновано удосконалений метод попередньої обробки сигналу ЕКГ для подальшого його використання у спайкінгових нейронних мережах; в роботі [9] – запропоновано удосконалену інформаційну технологію для класифікації сигналів ЕКГ на основі спайкінгових нейромереж, в роботі [10] – розроблено комп'ютерну програму класифікації часових рядів на основі спайкінгових нейромереж, що призначена для класифікації сигналів ЕКГ, які є окремим випадком часових рядів.

Апробація матеріалів дисертації.

Викладені в дисертації результати досліджень були апробовані на таких наукових конференціях: «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця; 1-а Міжнародна науково-практична конференція «Інформаційні системи та технології: результати і перспективи (IST 2024)», Київ; 1-а Міжнародна науково-практична конференція «Science, Technology and Industry in the Digital Age», Гамбург, Німеччина. September 24-26, 2025, Innovative Research in Science and Economy: Collection of Scientific Papers with Proceedings of the 4th International Scientific and Practical Conference. International Scientific Unity. Брюссель, Бельгія. Квітень 1-3, 2026.

Публікації.

За темою дисертації опубліковано 10 наукових робіт, у тому числі 5 статей [1], [2], [3], [4], [5] у наукових виданнях України, які входять до переліку фахових видань категорії Б, затверджених МОН України; 4 тез доповідей на конференціях [6], [7], [8], [9] та 1 свідоцтво про реєстрацію авторського права на твір (комп'ютерну програму) [10].

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, 4 розділів, загальних висновків, списку використаних джерел, додатків. Загальний обсяг дисертації становить 184 сторінки, з яких основний зміст викладений на 134 сторінках друкованого тексту, містить 40 рисунків, 14 таблиць. Список використаних джерел складається зі 143 найменувань. Додатки містять акти впровадження результатів роботи та лістинг програми.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ

1.1 Актуальність дослідження та сфери застосування інформаційних технологій ресурсоефективної реальночасової класифікації сигналів ЕКГ

Серцево-судинні захворювання (ССЗ) є однією з найбільших глобальних загроз здоров'ю людства, спричиняючи мільйони смертей щорічно. За даними Всесвітньої організації охорони здоров'я (ВОЗ), серцево-судинні захворювання є причиною понад 17,9 млн смертей щорічно, що робить їх провідною загрозою глобальному здоров'ю [11]. У США, наприклад, щорічно від ССЗ помирає близько 659 000 людей, а витрати на лікування та втрати продуктивності сягають 363 млрд доларів на рік [12]. Ці захворювання включають широкий спектр патологій, таких як аритмії, ішемічна хвороба серця, інфаркти міокарда та серцева недостатність. Особливу увагу привертають аритмії – порушення серцевого ритму, які можуть бути передвісниками серйозних ускладнень, включаючи раптову серцеву смерть.

Електрокардіографія (ЕКГ) є одним з найбільш доступних і неінвазійних методів діагностики серцевих аномалій. ЕКГ реєструє електричну активність серця, дозволяючи виявляти відхилення, такі як позачергові шлуночкові скорочення (ventricular ectopic beats), надшлуночкові екстрасистоли (supraventricular ectopic beats) та фібриляція передсердь (atrial fibrillation). Ці аномалії часто проявляються в змінах морфології хвиль Р, QRS-комплексу та Т-хвилі [13]. Традиційний підхід до аналізу ЕКГ базується на візуальному огляді кардіологами, який ефективний для короточасних записів, але стає неефективним для постійного моніторингу. Аритмії часто є інтермітуючими, тобто проявляються епізодично, і можуть залишитися непоміченими під час

разового візиту до лікаря [14]. Раннє виявлення аритмій може врятувати життя, оскільки дозволяє вчасно розпочати лікування.

Актуальність дослідження полягає в тому, що традиційні методи не забезпечують реального часу (real-time) моніторингу в повсякденному житті. Зростання чисельності населення похилого віку та поширення факторів ризику, таких як ожиріння, куріння та стрес, посилюють потребу в автоматизованих системах класифікації ЕКГ-сигналів. Інформаційні технології, зокрема штучний інтелект, дозволяють автоматизувати процес виявлення аномалій, роблячи діагностику більш доступною та ефективною [15], [16], [17], [18], [19], [20], [21], [22]. Однак, ключовим викликом є забезпечення ресурсоефективності – низького споживання енергії, пам'яті та обчислювальних ресурсів – для інтеграції в портативні пристрої.

Сучасні системи класифікації ЕКГ-сигналів стикаються зі значними викликами, пов'язаними з обчислювальною складністю та енергоспоживанням. Традиційні глибокі нейронні мережі (deep neural networks, DNN), такі як згорткові нейронні мережі (convolutional neural network, CNN) та рекурентні нейронні мережі (recurrent neural network, RNN), демонструють високу точність у класифікації аритмій – до 98% [23] на датасетах типу MIT-BIH. Проте, їхня висока обчислювальна складність робить їх непридатними для носимих пристроїв (wearable devices), таких як смарт-годинники чи імплантовані петльові реєстратори (ILR). Наприклад, у дослідженні [24] зазначено, що енергоспоживання традиційних CNN становить близько 0.077 Вт на інференс, що є критичним для батарейних пристроїв з обмеженим ресурсом.

Спайкінгові нейронні мережі (spiking neural networks, SNN) представляють третє покоління нейронних мереж, натхненних біологічними процесами мозку, де інформація передається через спайки (spike) з урахуванням часових аспектів. SNN дозволяють знизити енергоспоживання в 100-1000 разів порівняно з DNN, оскільки обчислення відбуваються лише під

час спайків, а не постійно [25]. У роботі [24] запропоновано SNN з енергоспоживанням 0.0074 від традиційних мереж, що робить їх ідеальними для локального обчислення – обчислень на краю мережі, без передачі даних у хмару. Це особливо актуально для реальночасової класифікації, де затримка повинна бути мінімальною (менше 1 секунди на серцебиття).

Пандемія COVID-19 посилила актуальність таких технологій, оскільки обмежила доступ до медичних закладів і стимулювала розвиток телемедицини. Під час пандемії зросла потреба в дистанційному моніторингу серцевих параметрів, оскільки COVID-19 часто спричиняє серцеві ускладнення, такі як аритмії [26]. Ресурсоефективні системи дозволяють інтегрувати класифікацію ЕКГ у мобільні додатки, де дані обробляються локально, забезпечуючи конфіденційність і незалежність від інтернету. Однак, виклики включають адаптацію до варіативності сигналів між пацієнтами (interpatient variability), шумів та обмежень апаратного забезпечення.

Дослідження показують, що SNN не тільки зменшують енергоспоживання, але й покращують адаптивність. Наприклад, у [13] SNN на базі leaky integrate-and-fire (LIF) нейронів досягають точності 93.8% на MIT-BIH датасеті з енергоспоживанням у кілька мікроват. Це контрастує з DNN, які вимагають потужних GPU для тренування та інференсу. Таким чином, ресурсоефективна реальночасова класифікація ЕКГ є ключовим напрямком для подолання бар'єрів у масовому впровадженні моніторингу ССЗ.

Інформаційні технології ресурсоефективної класифікації ЕКГ знаходять широке застосування в різних сферах, від повсякденного моніторингу до клінічної діагностики.

Перша сфера – це носимі пристрої, такі як смарт-годинники (наприклад, Apple Watch) та фітнес-браслети. Ці пристрої дозволяють щоденний моніторинг серцевого ритму без відвідування лікаря. У дослідженні [27] запропоновано SparrowSNN – апаратно-програмний засіб з енергоспоживанням 31.39 нДж на інференс і потужністю 6.1 мкВт, що

забезпечує точність 96.72% на MIT-BIH датасеті. Такі системи дозволяють виявляти аритмії в реальному часі, сповіщаючи користувача про потенційну небезпеку. Ринок носимих пристроїв стрімко зростає: за даними Statista [28], обсяг ринку носимих пристроїв досягне 93 млрд доларів до кінця 2026 року

Другою сферою є імплантовані пристрої, такі як кардіостимулятори та імплантовані петльові реєстратори (ILR). Ці пристрої призначені для пацієнтів з високим ризиком аритмій, де постійний моніторинг критичний. У [29] описано SNN для пристроїв з локальним обчисленням з низьким енергоспоживанням, що дозволяє класифікувати сигнали ЕКГ безпосередньо в пристрої. Це зменшує потребу в частій заміні елемента живлення і підвищує автономність.

Третя сфера – телемедицина та IoT-системи – ще одна ключова сфера. Інтеграція з мобільними додатками дозволяє передавати дані лікарям для раннього сповіщення. Наприклад, у [12] запропоновано стохастичну спайкінгову резервуарну мережу для класифікації з точністю 96.91%, що інтегрується в IoT для реального часу. Це особливо корисно в сільських районах або під час пандемій.

У клінічних системах автоматизована класифікація скорочує час аналізу в лікарнях. У [30] VPSNN досягає точності 97.16% для виявлення шлуночкових ударів (ventricular ectopic beats, VEB) з мінімальною кількістю параметрів (242), що полегшує інтеграцію в медичне обладнання.

З огляду на вище сказане, виникає потреба у розробці інформаційної технології класифікації кардіограм, що збільшить не тільки точність, але і чутливість та специфічність класифікації ЕКГ, що є ключовим показником у медичній сфері та у діагностуванні хвороб і при цьому буде мати високу ресурсоефективність та високу швидкодію (працюватиме в реальному часі).

1.2 Аналіз існуючих методів та інформаційних технологій класифікації сигналів ЕКГ

На рисунку 1.1 представлена узагальнена структура відомих систем класифікації електрокардіограм (ЕКГ). Процес починається з того, що кардіограф реєструє серцевий ритм пацієнта у формі ЕКГ-сигналу. Далі відбувається сегментація, під час якої виділяється один повний період серцевого скорочення. Цей сегмент проходить етап попередньої обробки, що включає усунення шумів, фону та нормалізацію даних. Нормалізація є ключовим кроком, оскільки вона забезпечує уніфікацію вхідних даних для подальшої ефективної обробки: у різних пацієнтів можуть суттєво відрізнятися амплітуди сигналу ЕКГ та тривалість серцевих циклів, що могло б ускладнити аналіз [17].

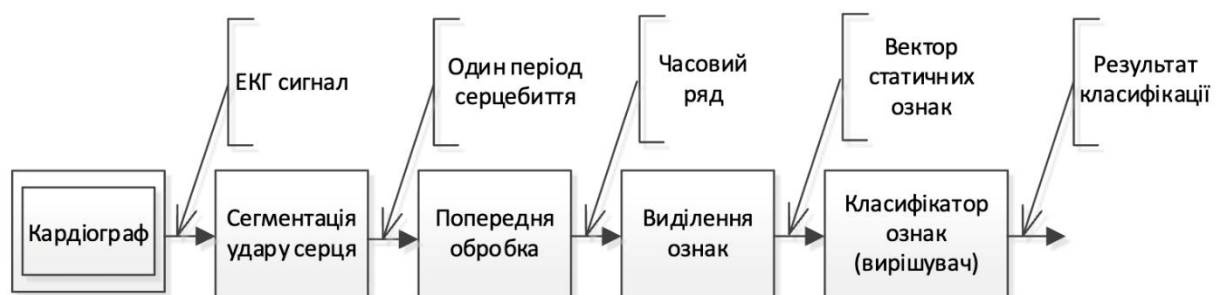


Рисунок 1.1 – Загальна схема класифікації електрокардіограм

Після цього з виділеного часового ряду $X(t)$ – тобто дискретних відліків оцифрованих амплітуд сигналу в конкретні моменти часу – витягуються різні ознаки, зокрема морфологічні. Традиційно ці ознаки добувалися евристичними методами, базуючись на припущеннях і накопиченому досвіді дослідників. Для зменшення розмірності та виділення найважливіших ознак застосовувалися такі перетворення, як перетворення Фур'є (FT) [31], вейвлет-перетворення (WT) [32], аналіз головних компонент (PCA) [33] та аналіз незалежних компонент (ICA) [34]. У результаті формується числовий вектор ознак, який подається на вхід класифікатора для виявлення серцевої аритмії чи

недостатності. Як класифікатори зазвичай використовуються методи машинного навчання, орієнтовані на ознаки з чіткими відмінностями для кожного типу аритмії, наприклад, метод опорних векторів (SVM), лінійний дискримінантний аналіз (LDA) та дерева рішень (DT).

Така традиційна схема класифікації ЕКГ має певні обмеження та недоліки. Головна проблема полягає в тому, що при перетворенні динамічного сигналу на набір статичних числових ознак втрачається значна частина цінної інформації про оригінальний сигнал, що може призвести до зниження точності класифікації. Крім того, ефективність усього процесу сильно залежить від вибору конкретного алгоритму для виділення ознак та методу класифікації, що робить систему вразливою до суб'єктивних факторів. Для подолання цих недоліків рекомендується проводити класифікацію безпосередньо на основі нормалізованих відліків сигналу у формі часових рядів, без їхнього перетворення на статичні ознаки. Це дозволяє зберегти динаміку даних. Водночас, для підвищення точності варто застосовувати класифікатори на базі штучних нейронних мереж, які краще справляються з обробкою складних патернів і можуть автоматично витягувати релевантні характеристики з сирих даних.

У ході аналітичного огляду літератури та досліджень було розроблено класифікаційну схему методів класифікації ЕКГ, яка акцентує увагу на автоматизованих підходах, оскільки класифікація за участю людини вважається менш перспективною через її суб'єктивність, витратність часу та обмежену масштабованість. Ця схема зображена на рисунку 1.2.

Автоматизована класифікація ЕКГ може бути систематизована за кількома ключовими ознаками, що дозволяють глибше зрозуміти різноманітність підходів:

1. Динамічні параметри. Основним вхідним параметром слугує вихідний сигнал кардіографа, представлений як функція часу $X(t)$. Цей параметр є фундаментальним для розпізнавання, оскільки він безпосередньо

відображає динаміку серцевої активності та містить усю первинну інформацію про ритм і форму хвиль.

2. Тип ознак часового ряду кардіограми. Для виявлення серцевої недостатності чи аритмії можуть використовуватися як окремі сегменти серцевих скорочень (для локального аналізу), так і обробка повної кардіограми в цілому (для глобального контексту). Це дозволяє адаптувати метод до конкретних завдань, наприклад, виявлення поодиноких аномалій чи довготривалих тенденцій.
3. Вид перетворення параметрів ЕКГ. Вхідні дані, подані у вигляді часових рядів, перетворюються за допомогою спеціалізованих алгоритмів виділення ознак. Серед найпоширеніших: перетворення Фур'є (FT) [31] для частотного аналізу, вейвлет-перетворення (WT) [32] для локалізованого часо-частотного розкладання, аналіз головних компонент (PCA) [33] для зменшення розмірності з збереженням варіації, аналіз незалежних компонент (ICA) [34] для розділення незалежних джерел сигналу, а також інші подібні методи, що оптимізують дані для подальшої класифікації.
4. Вид класифікатора. Після етапу виділення ознак розпізнавання здійснюється за допомогою класичних чи сучасних методів класифікації. До традиційних належать метод опорних векторів (SVM), лінійний дискримінантний аналіз (LDA) [35], дерева рішень (DT) та метод К-найближчих сусідів [36]. Для досягнення кращих результатів у складних задачах застосовуються штучні нейронні мережі, зокрема сучасні архітектури: спайкінгові нейронні мережі [37], [38], згорткові нейронні мережі [39] для обробки зображень чи сигналів, рекурентні нейронні мережі [40] для послідовних даних.

Найбільш перспективними вважаються спайкінгові нейронні мережі, оскільки вони спеціально адаптовані для прямого опрацювання часових рядів без попереднього перетворення у набір ознак та ефективно працюють з динамічними сигналами, імітуючи біологічні процеси в мозку.

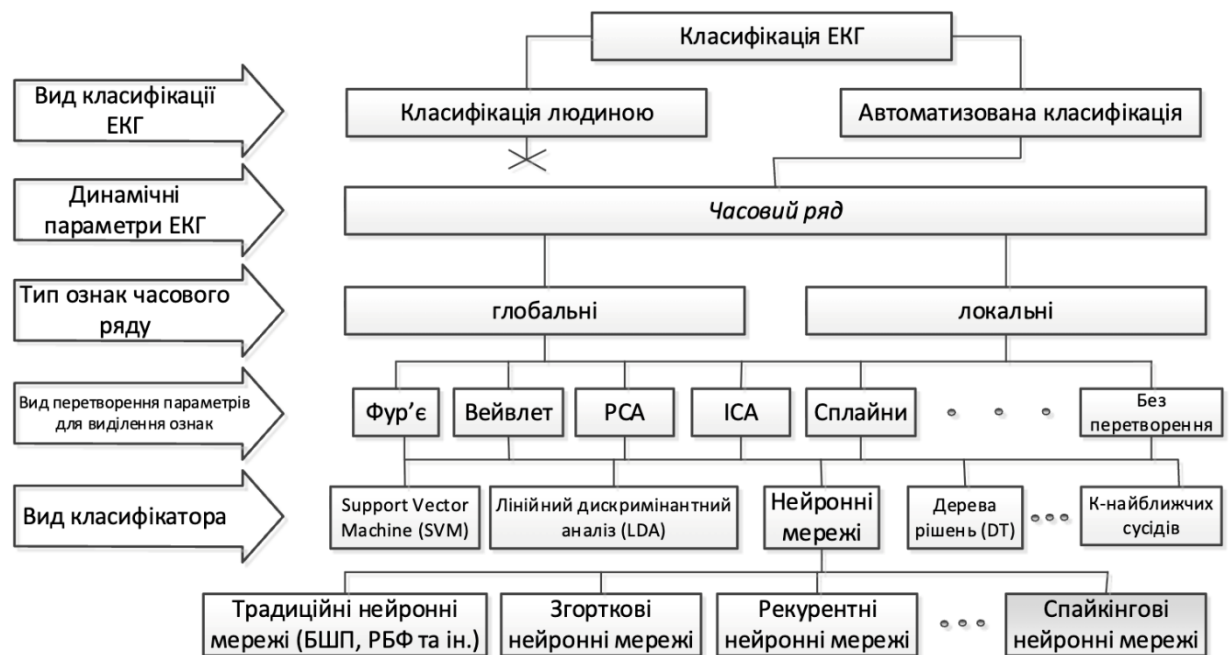


Рисунок 1.2 – Класифікаційна схема методів класифікації ЕКГ

Зокрема, для класифікації ЕКГ на портативних пристроях, таких як носимі монітори серцевого ритму чи мобільні кардіографи, SNN демонструють значні переваги, пов'язані з їхньою енергоефективністю: спайки генеруються лише в моменти активності, що мінімізує обчислювальні витрати та дозволяє працювати на пристроях з обмеженим живленням від батареї, досягаючи ультра-низького споживання енергії. Це робить їх ідеальними для тривалого моніторингу в реальному часі без частої підзарядки, на відміну від традиційних нейронних мереж, які вимагають постійних обчислень. Крім того, SNN забезпечують високу точність класифікації аритмій (наприклад, до 96,7% на датасетах MIT-BIH [41], з чутливістю понад 94% і F1-оцінкою близько 89% [24]), завдяки здатності обробляти темпоральні патерни сигналу безпосередньо, без втрати інформації через перетворення. Їхня сумісність з нейроморфним апаратним забезпеченням, таким як чіпи типу Intel Loihi [42], сприяє швидкій обробці даних у реальному часі, що критично для раннього виявлення аномалій серцевого ритму на портативних гаджетах, покращуючи загальну доступність і надійність медичного моніторингу в повсякденному житті. Водночас, SNN стійкі до шумів і варіацій сигналу, що часто

зустрічаються в мобільних умовах, і можуть інтегруватися з іншими біомедичними застосуваннями, роблячи їх перспективним рішенням для низькоенергетичних медичних систем.

Окрім чистих спайкінгових нейронних мереж, значний інтерес у контексті класифікації ЕКГ викликають гібридні моделі, які інтегрують спайкінгові нейронні мережі з іншими архітектурами, такими як згорткові нейронні мережі, трансформери чи традиційні штучні нейронні мережі, з метою поєднання біологічної правдоподібності SNN з потужними можливостями обробки даних інших методів [43]. Такі гібридні підходи, наприклад, моделі на основі спайкінгової уваги з трансформерами, дозволяють ефективно захоплювати як локальні темпоральні патерни, так і глобальні залежності в сигналах, досягаючи високої точності класифікації – до 95% для виявлення тимчасових зупинок дихання за час сну на основі одноканального ЕКГ та 88.4% для класифікації моторних уявлень на EEG-даних, з F1-оцінкою понад 0.86 [44]. Інші приклади включають DNN-SNN конверсію, де ваги тренованої глибокої нейронної мережі переносяться в SNN для створення гібридної системи, що забезпечує точність до 97.8% на датасетах MIT-BIH для п'яти класів аритмій, зберігаючи при цьому подію-керовану обробку для зменшення обчислювальних витрат [45]. Переваги таких моделей полягають у покращеній узагальненості, стійкості до шуму та варіацій сигналу, що часто зустрічаються в реальних умовах моніторингу, а також у можливості інтеграції з нейроморфним апаратним забезпеченням, таким як чіпи Intel Loihi, для реального часу обробки на портативних пристроях [46]. Це робить гібридні SNN особливо перспективними для біомедичних застосувань, де потрібен баланс між точністю та ефективністю, наприклад, у носимих кардіомоніторах для тривалого спостереження за серцевим ритмом без частого заміни батареї [47], [48].

Проте, гібридні моделі не позбавлені недоліків, зокрема, вони часто демонструють дещо вище енергоспоживання порівняно з чистими SNN через додавання компонентів на кшталт ANN або трансформерів, які вимагають

більше параметрів та обчислень, що може сягати 0.42 мДж на удар серця на апаратному рівні, проти мікроджоулів чи наноджоулів для оптимізованих SNN. Крім того, ресурсоефективне онлайн-навчання на портативних пристроях без доступу до хмарних обчислень або оновлень є проблематичним: хоча STDP (спайк-часова залежна пластичність) у SNN дозволяє адаптацію без вчителя в реальному часі, гібридні системи часто залежать від офлайн-тренінгу або конверсії, що ускладнює динамічне донавчання на пристроях з обмеженими ресурсами, призводячи до потенційної втрати точності через відсутність потужних GPU чи хмарних серверів [42]. Це обмежує їхню застосовність у повністю автономних сценаріях, таких як віддалений моніторинг у регіонах з поганим інтернет-з'єднанням, де потрібна локальна адаптація моделі до індивідуальних варіацій пацієнта без зовнішньої підтримки.

Отже, в результаті огляду існуючих методів та підходів для класифікації сигналів ЕКГ було визначено більш перспективним використання спайкінгових нейронних мереж, оскільки вони мають кращу ресурсоефективність порівняно як з традиційними (згортковими, рекурентними і тд.) так і з новими гібридними мережами.

1.3 Аналіз існуючих методів та інформаційних технологій класифікації сигналів ЕКГ на основі спайкінгових нейронних мереж

Спайкінгові нейронні мережі (Spiking Neural Networks, SNN) представляють собою третє покоління нейронних мереж, яке відрізняється від традиційних штучних нейронних мереж (Artificial Neural Networks, ANN) та мереж другого покоління, таких як перцептрони чи сигмоїдні мережі, за рахунок імітації біологічних процесів у мозку [23]. На відміну від ANN [49], [50], де інформація обробляється у формі безперервних значень активацій, SNN працюють на основі дискретних імпульсів (спайків), що генеруються

нейронами в залежності від часу. Цей підхід робить SNN більш біологічно правдоподібними, оскільки вони відтворюють механізми синаптичної пластичності та часового кодування сигналів, характерні для нервової системи людини.

Одним з ключових механізмів, що лежить в основі SNN, є пластичність, залежна від часу спайків (Spike-Timing Dependent Plasticity, STDP). Згідно з [23], STDP є біологічно натхненим правилом навчання, де ваги синапсів змінюються залежно від відносного часу пре- та постсинаптичних спайків. Якщо пресинаптичний спайк передує постсинаптичному, вага посилюється (потенціація), а в протилежному випадку – послаблюється (депресія). Це дозволяє SNN адаптуватися до динамічних сигналів, таких як ЕКГ, де часові характеристики (наприклад, інтервали між піками QRS-комплексу) є критичними для класифікації аритмій. Крім того, R-STDP (Reward-Modulated STDP) додає елемент винагороди чи покарання, що робить навчання більш ефективним для задач класифікації, дозволяючи мережі фокусуватися на релевантних паттернах [23].

Переваги SNN над традиційними мережами особливо помітні в контексті класифікації сигналів ЕКГ. По-перше, низьке енергоспоживання: SNN споживають енергію лише під час генерації спайків, що робить їх ідеальними для портативних пристроїв з обмеженими ресурсами. Наприклад, у [23] показано, що енергоспоживання SNN для класифікації ЕКГ становить близько 1.78 μJ на удар серця, що на 2-9 порядків менше, ніж у DNN (Deep Neural Networks). Це досягається завдяки асинхронній обробці даних, де нейрони активуються тільки за потреби, на відміну від синхронної обробки в ANN, де всі шари обчислюються щоразу. По-друге, біологічна подібність: SNN імітують нейрони мозку, такі як моделі Leaky Integrate-and-Fire (LIF), де мембранний потенціал накопичується з витокom, подібно до реальних нейронів [3]. Це дозволяє краще обробляти часові залежності в ЕКГ-сигналах, таких як варіації ритму, що є ключовими для виявлення аритмій, як вказано в [2].

У контексті медичних застосувань, SNN вирішують проблеми традиційних методів класифікації ЕКГ, таких як морфологічний аналіз (наприклад, QRS-виявлення за допомогою фільтрів [51]) чи DNN (наприклад, CNN для екстракції ознак [52]). Традиційні методи часто вимагають ручного виділення ознак, що робить їх чутливими до варіацій форми ЕКГ у різних пацієнтів (inter-patient), тоді як DNN мають високу обчислювальну складність, непридатну для носимих пристроїв [53]. SNN, навпаки, поєднують автоматичне навчання з низьким енергоспоживанням, що критично для реального часу моніторингу серцевих ритмів на носимих пристроях, як описано в [54]. Згідно з [24], SNN досягають точності, близької до DNN (наприклад, 96.91% для класифікації аритмій), але з меншим споживанням ресурсів.

Крім того, SNN дозволяють інтеграцію з нейроморфними чипами, такими як Intel Loihi [42] чи Brainchip Akida [55], де апаратна реалізація спайків забезпечує ультра-низьке енергоспоживання [23]. Це відкриває шлях до автономних систем для раннього виявлення серцево-судинних захворювань, які, за даними WHO [56], є причиною 17.9 млн смертей щорічно. Біологічна подібність SNN також полегшує інтерпретацію результатів, оскільки ваги мереж можуть відповідати фізіологічним параметрам, як позиції та ширини хвиль P-QRS-T у ЕКГ [57], [58].

У цьому розділі ми зосередимося на аналізі існуючих методів на основі SNN для класифікації ЕКГ, їхніх архітектур та порівнянні результатів. Причому ми не будемо розглядати гібридні технології [24] на основі SNN, де використовується перетворення ANN→SNN або CNN→SNN, а проаналізуємо тільки методи, де використовується тільки чисто SNN без перетворення в інші ANN. Методи на основі чисто SNN є більш перспективними, оскільки не передбачають складних та інформаційно неадекватних процедур перетворення інших навчених мереж у навчену SNN.

Проаналізуємо знайдені джерела [12], [25], [29], [59], [60], [61] – 6 джерел, де описано методи класифікації ЕКГ на основі чисто SNN, де і

навчання проводиться для SNN. Загалом, SNN представляють перспективний напрямок для енергоефективної класифікації ЕКГ, поєднуючи біологічну правдоподібність з практичними перевагами для медичних пристроїв.

Класифікаційна схема методів класифікації сигналів ЕКГ на основі саме SNN представлена на рис. 1.3.

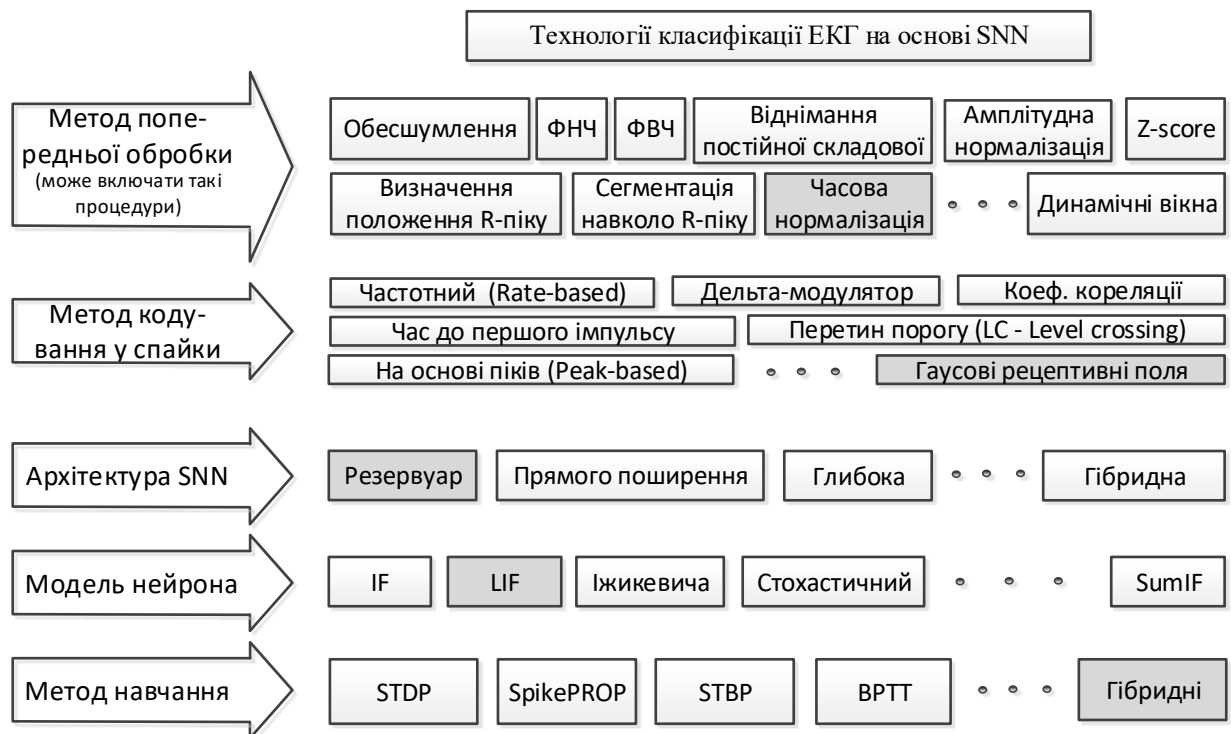


Рисунок 1.3 – Класифікаційна схема методів класифікації сигналів ЕКГ на основі SNN

Основні класифікаційні ознаки такі:

- Метод попередньої обробки сигналу ЕКГ;
- Метод кодування відліків ЕКГ у спайки;
- Архітектура спайкінгової нейронної мережі;
- Модель нейрона;
- Метод навчання.

У процесі класифікації електрокардіограм (ЕКГ) за допомогою нейронних мереж, зокрема спайкінгових нейронних мереж (SNN), попередня обробка сигналу відіграє критичну роль. Вона спрямована на усунення шумів,

нормалізацію даних та підготовку сигналу до введення в мережу, що дозволяє покращити точність класифікації та зменшити обчислювальні витрати. У літературних джерелах, які розглядаються в цій роботі, методи попередньої обробки варіюються від мінімальних (використання сирих даних) до складних (фільтрація, сегментація та кодування в спайки). Нижче наведено аналіз методів та їх порівняння для кожного з шести джерел.

A Novel Spiking Neural Network for ECG signal Classification. Amrita Rana та Kyung Ki Kim (2021) [59].

Робота [59] присвячена вирішенню проблеми діагностики серцево-судинних захворювань за допомогою аналізу сигналів електрокардіограми (ЕКГ) на пристроях, що носяться. Автори зазначають, що сучасні моделі глибокого навчання мають високу точність, але потребують надмірних обчислювальних ресурсів, пам'яті та енергії, що робить їх важкими для розгортання в обмежених вбудованих системах. Для подолання цього запропоновано використання спайкінгових нейронних мереж, які імітують біологічні процеси та працюють на основі розріджених імпульсів.

Технологічне рішення: Основною інновацією роботи є розробка бінаризованої SNN. У цій моделі синаптичні ваги обмежені лише двома значеннями: +1 та -1. Такий підхід дозволяє замінити складні та енергозатратні операції множення матриць на просте додавання та віднімання, що важливо для апаратної реалізації на FPGA чи спеціалізованих чипах.

Основні характеристики системи:

- Метод обробки: Вхідні дані ЕКГ перетворюються на 2D-зображення, які потім кодуються у послідовності спайків за допомогою імовірнісного методу (Probabilistic encoding).
- Архітектура: Мережа складається з 3 шарів: вхідного - 784 нейрони, що відповідають розміру зображення, прихованого - 64 нейрони та вихідного - 3 нейрони для класифікації типів сигналів (рис. 1.4).
- Метод навчання – STDP.
- Апаратна база: Нейронну мережу було розгорнуто та протестовано на

платі FPGA PYNQ-Z2.

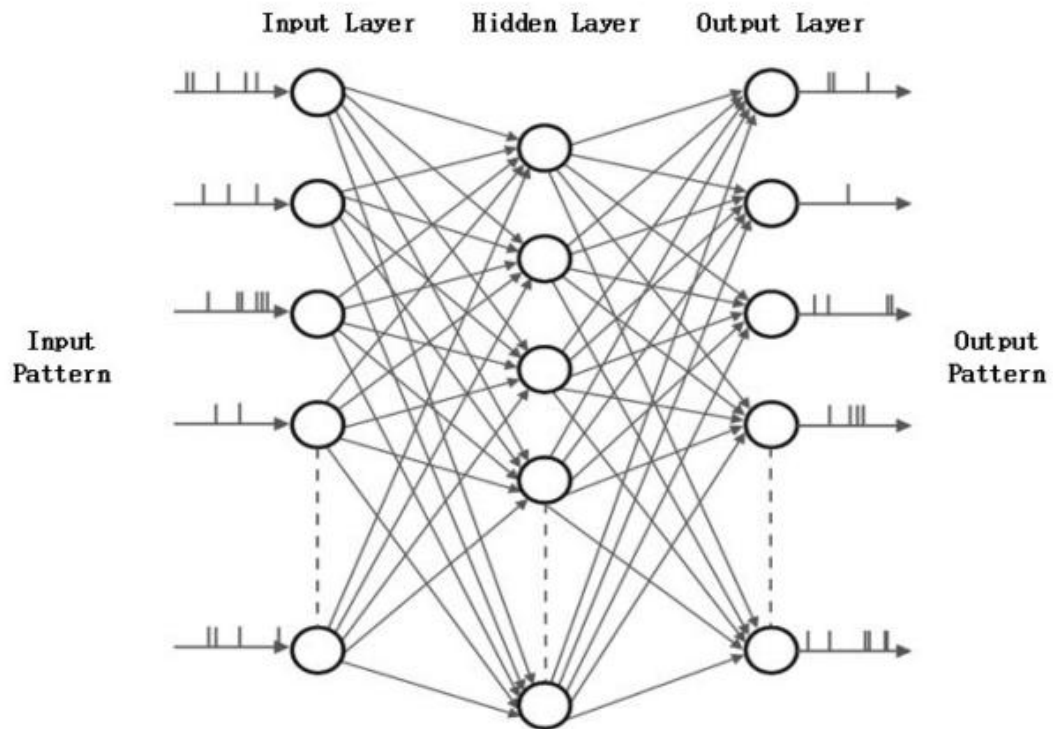


Рисунок 1.4 – Архітектура нейронної мережі [59]

Згідно з результатами моделювання, запропонована бінаризована SNN продемонструвала точність класифікації на рівні 85.0%. Хоча це дещо менше, ніж у повнорозмірних CNN (98.1%), бінаризована модель виявилася значно ефективнішою за показниками ресурсів:

- Енергоспоживання: становить лише 0.176 Вт, що майже в 9 разів менше за традиційну SNN (1.55 Вт).
- Використання пам'яті: модель займає лише 48% ресурсів пам'яті порівняно з 98% у звичайної SNN та 350% у CNN (що робило CNN неможливою для запуску безпосередньо на FPGA без оптимізації).

A Neuromorphic Processing System With Spike-Driven SNN Processor for Wearable ECG Classification. Haoming Chu, Yulong Yan та інші (2022) [60].

Дослідження [60] спрямоване на створення енергоефективної системи для безперервного моніторингу ЕКГ у пристроях, що носяться. Традиційні методи класифікації на основі глибоких нейронних мереж забезпечують

високу точність, але споживають занадто багато енергії для автономних датчиків. Автори пропонують повністю подієво-керовану систему, яка використовує спайкінгові нейронні мережі та спеціалізований процесор для обробки сигналів безпосередньо на пристрої.

Ключовим моментом рішення системи є використання дискретизації за перетином рівнів (Level Crossing sampling). Така дискретизація генерує спайк лише тоді, коли сигнал змінюється на певну величину, що забезпечує неспотворене часове кодування сигналу та стискає обсяг даних у 55,9 рази. Це спрощує архітектуру системи, оскільки дані подаються в SNN у 1-бітному представленні без додаткового складного кодування.

Архітектура (рис. 1.5) та навчання:

- **Модель:** Використано спайкінговий рекурентний багатошаровий перцептрон (rMLP) з 271 нейроном моделі Leaky Integrate-and-Fire. Архітектура включає вхідний шар з 96 нейронів, «рідкий пул» для виділення часових ознак з 120 нейронів, прихований шар з 50 нейронів та вихідний шар класифікації з 5 нейронів. Було використано неповну зв'язність між вхідним, резервуаром та прихованими шарами.

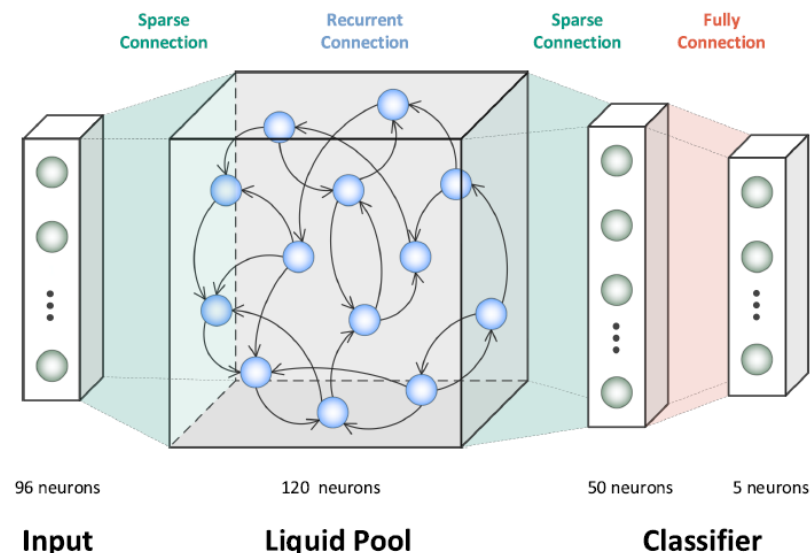


Рисунок 1.5 – Архітектура спайкінгової нейронної мережі rMLP [60]

- **Метод навчання:** Застосовано покращений алгоритм просторово-

часового зворотного поширення помилки (STBP) з апаратною оптимізацією. Оптимізація включає мінімізацію частоти спрацьовування нейронів, видалення зайвих зв'язків та 6-бітове квантування ваг.

- Апаратна реалізація: Розроблено спеціалізований ASIC-процесор (технологія 40 нм CMOS) з ієрархічною схемою доступу до пам'яті, що зменшує потребу в SRAM більш ніж в 11 разів.

Система продемонструвала високу продуктивність на базі даних MIT-BIH:

- Точність (Ассигасу): становить близько 98% для класифікації 5 категорій ударів серця.

- Енергоспоживання: 0,75 мкДж на одну класифікацію (при частоті роботи чіпа 100 МГц), що відповідає середній потужності 0,93 мкВт.

An SNN Based ECG Classifier For Wearable Edge Devices. Dighanchal Banerjee, Sounak Dey, Arpan Pal [29].

Дослідження присвячене розробці систем моніторингу ЕКГ у реальному часі для портативних та імплантованих пристроїв, таких як розумні годинники та кардіостимулятори. Автори підкреслюють, що існуючі методи глибокого навчання є занадто ресурсномісткими для пристроїв з обмеженим об'ємом пам'яті та зарядом батареї. Як рішення пропонується використання спайкінгових нейронних мереж у поєднанні з нейроморфними обчислювальними платформами, що забезпечує асинхронну обробку даних і низьке енергоспоживання.

Головним внеском роботи є пропозиція нового методу кодування – Peak-based encoder (кодування на основі піків). На відміну від традиційних методів, цей кодувальник фокусується лише на морфологічно значущих компонентах ЕКГ: хвилях Р, комплексах QRS та хвилях Т.

Автори порівняли дві архітектури (рис 1.6):

- SNN:Reservoir (Резервуарна мережа): Використовує рекурентні зв'язки для виділення просторово-часових ознак.

- Feed-forward (Мережа прямого поширення): Простіша топологія, де

навчання відбувається за правилом STDP (пластичність, залежна від часу спайку). В обох випадках використано нейрони моделі LIF та вихідний класифікатор на основі логістичної регресії.

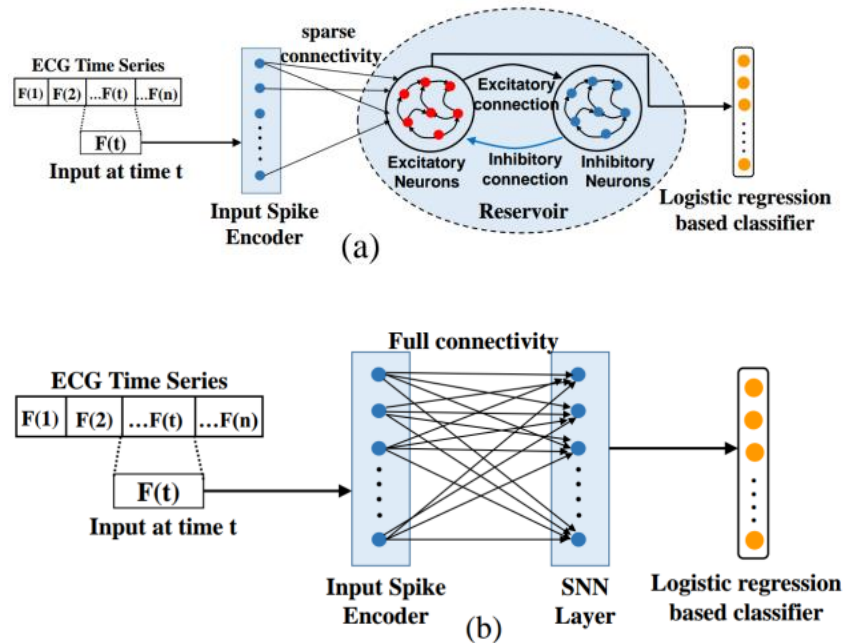


Рисунок 1.6 – Розглянуті в [29] архітектури SNN:

а) резервуарна SNN; б) SNN прямого поширення

Система була протестована на п'яти наборах даних (включно з MIT-BIH та PhysioNet 2017). Основні результати:

- Найкраща комбінація: Поєднання Peak-based та Feed-forward мережі показало найвищу точність на всіх датасетах. На базі MIT-BIH точність склала 94,3%.
- Енергоефективність: Оцінювалася через кількість синаптичних операцій (SOP) на часовий крок. Для MIT-BIH цей показник склав лише 33 SOP, що є найменшим значенням серед усіх протестованих конфігурацій. Дослідження доводить, що фокус на морфології сигналу замість кодування кожного окремого значення амплітуди дозволяє значно зменшити обчислювальні витрати без критичної втрати точності.

Neuromorphic Computing Based on Stochastic Spiking Reservoir for

Heartbeat Classification. Chia Yee Saw ma Yan Chiew Wong (2022) [12].

Дослідження спрямоване на розв'язання проблеми високої обчислювальної вартості класифікації ЕКГ-сигналів у реальному часі. Автори зазначають, що більшість існуючих систем потребують видалення значної кількості ознак сигналу, що веде до надмірних витрат апаратних ресурсів. Основна мета роботи – створити систему, яка була б енергоефективною, компактною для реалізації на залізі та здатною адаптуватися до індивідуальних особливостей серцевого ритму різних пацієнтів.

Автори пропонують стохастичну спайкінгову нейронну мережу, яка базується на архітектурі резервуарних обчислень (рис. 1.7).

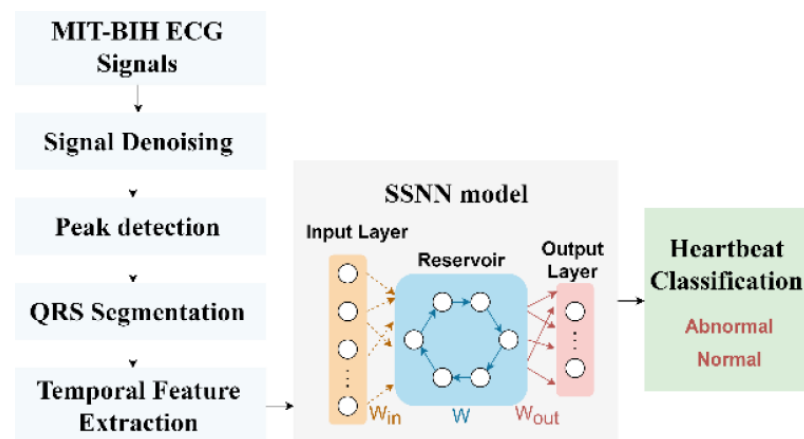


Рисунок 1.7 – Схематичне представлення класифікатора ЕКГ на основі SNN [12]

Основними нововведеннями є:

- Узгодження за шаблоном: Для подолання проблеми індивідуальної варіабельності форми сигналів застосовано метод порівняння сегментованого удару з класовим шаблоном за допомогою нормованого коефіцієнта кореляції.
- Стохастичні обчислення (SC): Це парадигма, де числа представляються у вигляді імовірнісних послідовностей бітів. Це дозволяє виконувати складні операції за допомогою надзвичайно простої логіки (елементи XNOR та MUX), що важливо для економії площі кристала.
- Топологія SCR (Simple Cycle Reservoir): Резервуар мережі

побудований за принципом простого циклу і складається лише з 20 стохастичних нейронів. Така структура мінімізує кількість синаптичних зв'язків.

Система була протестована на базі даних MIT-BIH (43 записи).

- Метрики ефективності: Досягнуто точності 95,10%, чутливості 97,41% та специфічності 93,68%.

- Апаратні ресурси: При розгортанні на FPGA Xilinx Zynq-7000 система використала лише 692 LUT (Look-Up Table, Таблиця підстановки) – 0,65% – та 440 FF (Flip-Flop, Тригер) – 0,83% – що є значно меншим показником порівняно з іншими типами нейронних мереж. Дослідження доводить, що стохастична природа обчислень у поєднанні з резервуарною топологією дозволяє створити високоефективне рішення для медичного обладнання.

ECG Classification Algorithm Based on STDP and R-STDP Neural Networks for Wearable Devices. Alireza Amirshahi та Matin Hashemi (2019) [25].

Робота присвячена створенню алгоритму класифікації ЕКГ у реальному часі для персональних пристроїв, що носяться. Автори зазначають, що традиційні методи забезпечують високу точність, але є занадто енергомісткими для невеликих пристроїв з обмеженим об'ємом батареї. Як рішення пропонується використання спайкінгових нейронних мереж, де інформація кодується в часі виникнення імпульсів, що дозволяє нейронам споживати енергію лише в моменти активності.

Головною особливістю роботи є запропонована багатошарова архітектура SNN (рис. 1.8), що поєднує різні механізми біологічного навчання:

- Кодування спайків: Сигнал розбивається на вікна, що перекриваються, і перетворюється на спайки за допомогою випадкового процесу Пуассона, де частота імпульсів пропорційна амплітуді сигналу.

- Гаусів шар: Коригує кількість спайків у вікнах, щоб посилити значущі піки та послабити шум.

- Шар STDP (некероване навчання): Використовує пластичність, залежну від часу спайку, для автоматичного виділення патернів аритмії без

використання міток класів. Автори оптимізували правила навчання STDP, зробивши механізм послаблення зв'язків сильнішим за посилення, що дозволило краще захоплювати складні морфологічні ознаки.

- Гальмівний шар: Запобігає тому, щоб усі нейрони реагували на один і той самий патерн, стимулюючи різноманітність виділених ознак.

- Шар R-STDP (кероване навчання): Використовує механізм винагороди та покарання (нагороджувальне STDP) для остаточної класифікації виділених ознак за категоріями.

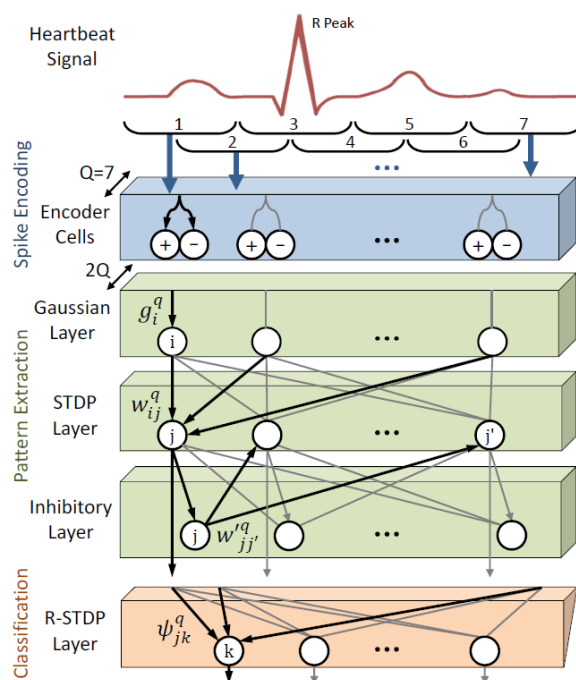


Рисунок 1.8 – Багатошарова архітектура SNN, запропонована в [25]

Система була протестована на базі даних MIT-BIH (група складних аритмій DS2) і продемонструвала:

- Високу точність: 97,9%, що є порівнянним із сучасними глибокими нейронними мережами (наприклад, 98,6% у моделей на основі CNN).

- Низьке енергоспоживання: Лише 1,78 мкДж на один удар серця, що значно менше за будь-які рішення на основі класичних нейронних мереж.

- Реальний час: Конфігурація підтримує обробку до 300 ударів на хвилину, що повністю задовольняє потреби медичного моніторингу.

SparrowSNN: A Hardware/software Co-design for Energy Efficient ECG Classification. Zhanglu Yan, Zhenyu Bai, Tulika Mitra and Weng-Fai Wong (2023) [61].

Робота присвячена розробці системи безперервного моніторингу серцевих захворювань, які часто є безсимптомними та потребують аналізу в реальному часі на портативних пристроях. Традиційні архітектури SNN, що використовують моделі Integrate-and-Fire або Leaky Integrate-and-Fire, мають суттєвий недолік при апаратній реалізації: вони потребують багаторазового доступу до пам'яті та повторюваних обчислень для кожного часового кроку, що нівелює їхню енергоефективність. Автори SparrowSNN пропонують переосмислити робочий процес SNN з апаратної точки зору, поєднуючи оптимізацію алгоритмів із розробкою спеціалізованого ASIC-чипа.

Ключовою інновацією є механізм активації Sum-Spikes-Fire (SSF). На відміну від стандартних моделей, SSF спочатку підраховує загальну кількість спайків у вхідній послідовності, а потім обчислює зважену суму за один прохід. Це дозволяє:

- Зменшити кількість звернень до пам'яті з T разів (де T – довжина часового вікна) до одного разу.
- Замінити послідовні додавання на одну високоефективну операцію множення з накопиченням (MAC).
- Підвищити стійкість моделі при малих часових вікнах ($T < 16$), де традиційні моделі IF часто втрачають точність через залишковий мембранний потенціал.

Архітектура та навчання:

- Тип мережі: Використано спайкінговий багатошаровий перцептрон, оскільки він потребує менше енергії на обчислення порівняно зі спайкінговими CNN при аналогічній точності для ЕКГ.
- Метод навчання: Застосовано концепцію навчання ANN з подальшою конвертацією в SNN. Для мінімізації втрати точності під час конвертації використовувалися функції Clamp-and-Quantize, які вирівнюють активації

ANN до дискретних значень SNN.

- Апаратна реалізація: Розроблено ASIC-дизайн (технологія 22 нм FD-SOI) з одним обчислювальним вузлом, що працює на частоті 4 МГц.

SparrowSNN продемонструвала найкращі результати серед порівнюваних систем:

- Точність (Accuracy): 96,72% на базі MIT-BIH після онлайн-навчання для конкретного пацієнта.

- Енергоспоживання: Рекордно низьке 31,39 нДж на інференс, потужність 6,1 мкВт.

Розглянемо методи попередньої обробки ЕКГ, які можна порівняти за критеріями: сегментація, нормалізація, кодування для SNN (див. табл. 1.1).

Сегментація. Аналіз показує два основні підходи: фіксовані часові вікна навколо R-піку та адаптивні методи, що враховують варіабельність серцевого ритму.

Фіксовані вікна навколо R-піку є найпоширенішим методом. У роботі [61] використовується вікно з 180 точок, центроване на R-піку (по 90 точок з кожного боку). Робота [25] формує сегмент тривалістю 0,7 с (0,25 с до та 0,45 с після R-піку), який далі ділиться на дрібні вікна, що перекриваються. Найбільш вузька сегментація у роботі [12] – лише 75 відліків навколо R-піку, що охоплюють виключно QRS-комплекс для мінімізації обчислень. Адаптивну сегментацію запропоновано в [60] для цього використано динамічну довжину сегмента, де межі визначаються як середина між поточним R-піком та сусідніми (попереднім і наступним). Ці дані потім приводяться до фіксованих 48 часових слотів через відсікання або заповнення нулями для подальшої обробки. У роботі [59] дані оброблюються як 2D-зображення (статичні картинки ЕКГ-сигналів), що відрізняє цей підхід від аналізу часових рядів.

Нормалізація. Методи нормалізації варіюються від простого масштабування значень до складних статистичних підходів. Масштабування діапазону: у [61] виконується нормалізація значень у діапазон $[0; 1]$; у

[25] амплітуда сигналу використовується для встановлення частоти генерації спайків, додаючи невелике зміщення. Статистична та порогова нормалізація: [12] застосовує нормований коефіцієнт кореляції для порівняння сегментованого удару з шаблоном класу, що дозволяє нівелювати індивідуальні відмінності пацієнтів. У [60] нормалізація інтегрована в процес АЦП, де встановлено фіксований інтервал рівнів 0,1 мВ для детекції перетинів, що фактично квантує амплітуду сигналу. У [59] нормалізація відбувається на рівні інтенсивності пікселів 2D-зображення перед їх перетворенням у спайки.

Таблиця 1.1 – Порівняння методів попередньої обробки

Джерело	Сегментація	Нормалізація	Кодування
<i>Amrita Rana та ін.</i> [59]	-	-	Метод імовірнісного кодування
<i>Haoming Chu та ін.</i> [60]	Динамічне вікно з перетворенням у 48 часових слотів	-	Дискретизація за перетином рівнів (LC-ADC)
<i>Dighanchal Banerjee та ін.</i> [29]	Фіксоване вікно 300 відліків	-	Розглянуто різні методи кодування. Найкращий - Peak-based encoder
<i>Chia Yee Saw та ін.</i> [12]	Вікно 75 відліків навколо R-піку	Використання нормованого коефіцієнта кореляції для узгодження з шаблоном	Стохастичне обчислення – перетворення бінарних чисел у імовірнісні послідовності бітів
<i>Alireza Amirshahi та ін.</i> [25]	0.25 с до R-піку та 0.45 с після нього	-	Спайкінгове кодування за допомогою випадкового процесу Пуассона
<i>Zhanglu Yan та ін.</i> [61]	Вікно 90 відліків навколо R-піку	[0;1]	Частотне кодування
Запропоноване рішення	Динамічне вікно	[0;1]	Гаусові рецептивні поля

Кодування для SNN. Це ключовий етап перетворення аналогового або цифрового сигналу ЕКГ у спайкінгову форму. Виділяється кілька стратегій:

- Частотне та ймовірнісне кодування: Пуассонівський процес: у роботі [25] використовується процес Пуассона, де частота спайків прямо пропорційна амплітуді сигналу, з окремими каналами для позитивних та негативних значень. Частотне кодування: [61] реалізує частотне кодування через механізм Integrate-and-Fire, де значення подається на спайкінговий шар протягом певного часу T . Імовірнісне кодування: [59] перетворює значення пікселів зображення ЕКГ у послідовності спайків.

- Часове та подієво-орієнтоване кодування: Level Crossing: у [60] реалізовано часове кодування, де спайки генеруються лише при перетині сигналом амплітудних порогів. Це стискає дані у 55,9 рази. Дельта-модулятор: [29] згадує цей метод як кодування змін значень між сусідніми кроками (спайк при перевищенні порога зміни).

- Морфологічне кодування: [29] пропонує унікальний Peak-based encoder, який використовує таймінги піків, початків та кінців хвиль P, QRS та T як моменти появи спайків. Це забезпечує максимальну розрідженість даних.

- Стохастичне кодування: [12] використовує B2P-конвертери для перетворення бінарних чисел у ймовірнісні бітові потоки, де значення кодується часткою одиниць у послідовності.

Таким чином, більшість праць не аналізує динамічну сегментацію сигналу ЕКГ, що призведе до неточностей у реальночасовій класифікації при збільшенні частоти пульсу, оскільки часове вікно буде включати в себе дані з наступних чи попередніх ударів. Також для збереження енергоефективності не застосовуються складні алгоритми попередніх обробок сигналів.

Розглянемо використовувані архітектури SNN, які можна порівняти за критеріями: топологія мережі, кількість шарів/нейронів, тип нейронів, метод навчання (табл. 1.2).

Таблиця 1.2 – Порівняння архітектур SNN

Джерело	Топологія мережі	Кількість шарів/нейронів	Тип нейронів	Метод навчання
<i>Amrita Rana та ін. [59]</i>	Прямого поширення	3 шари 784-64-3	-	STDP
<i>Haoming Chu та ін. [60]</i>	Гібридна (прямого поширення + «резервуарна»)	4 шари 96-120-50-5	LIF	Апаратно-орієнтоване просторово-часове зворотне поширення помилки зі зниженням частоти спайків (G-STBP)
<i>Dighanchal Banerjee та ін. [29]</i>	Досліджено 2 варіанти: прямого поширення і «резервуарна»	1000-3000 нейронів залежно від кодувальника	LIF	STDP
<i>Chia Yee Saw та ін. [12]</i>	Стохастична спайкінгова нейронна мережа з топологією простого циклічного резервуара	20 стохастичних нейронів у циклічній топології	Стохастичні сигмоїдальні юніти, змодельовані як спайкінгові нейрони	Узгодження шаблону за допомогою коефіцієнта кореляції. Ваги зчитування навчаються лінійною регресією
<i>Alireza Amirshahi та ін. [25]</i>	Прямого поширення	4 шари	LIF	STDP, R-STDP
<i>Zhanglu Yan та ін. [61]</i>	Прямого поширення	5 шарів 180-56-56-56-4	Sum-Spikes-Fire	Back Propagation
<i>Запропоноване рішення</i>	Резервуарна	3 шари 50-2048-12 з розрідженими зв'язками між шарами	LIF	3-етапне навчання

Аналіз порівняння:

Топологія мережі. Аналіз робіт дозволяє виділити три основні підходи

до організації топології мережі, кожен з яких має свої переваги для аналізу біосигналів:

- **Прямого поширення:** Це найбільш затребуваний тип топології для портативної діагностики завдяки простоті реалізації та передбачуваності обчислень. Прикладами є 3-рівнева бінаризована мережа (784-64-3) [59], 4-шарова структура для послідовного вилучення ознак і класифікації [25], а також 5-рівневий SMLP проекту SparrowSNN [61].

- **Рекурентна топологія:** Використовується для захоплення складних часових залежностей ЕКГ-сигналу. У джерелах описано модель спайкінгового перцептрону, що містить спеціальний рекурентний шар на 120 нейронів із розрідженими внутрішніми зв'язками, що дозволяє мережі мати «пам'ять» про попередні стани сигналу [60].

- **Резервуарна топологія (Reservoir computing) [62]:** Базується на використанні фіксованого «резервуара» нейронів. Дослідження охоплюють як масивні популяції нейронів (до 4000 одиниць) для вилучення просторово-часових ознак [29], так і надкомпактні циклічні топології [59], що складаються всього з 20 нейронів, з'єднаних у просте коло для мінімізації апаратних витрат.

Розглянемо використовувані типи моделей нейрону. Вибір моделі нейрона критично впливає на баланс між точністю та складністю апаратної частини:

- **LIF (Leaky Integrate-and-Fire):** Найбільш вживана модель [25], [29], [60], що імітує накопичення потенціалу мембрани з поступовим «витоком». Вона забезпечує високу точність [60] при відносно простих обчисленнях.

- **SSF (Sum-Spikes-Fire):** Інноваційна модель, запропонована для проекту SparrowSNN [61], яка роз'єднує доступ до пам'яті та процес генерації спайків. Вона спочатку підраховує загальну кількість вхідних спайків, а потім розраховує мембранний потенціал за один цикл звернення до пам'яті, що значно швидше за традиційні моделі.

- **Стохастичні нейрони:** Представлені як сигмоїдальні юніти, що оперують імовірнісними бітовими потоками [12]. Арифметичні операції

(множення, додавання) у таких нейронах виконуються за допомогою XNOR та MUX.

Розглянемо використовувані методи навчання, які варіюються від біологічно правдоподібних правил до глибокої оптимізації під конкретне апаратне забезпечення:

- STDP (Spike-timing dependent plasticity): [25], [29], [59] Навчання без нагляду, за якого ваги синапсів змінюються залежно від мілісекундної різниці в часі між вхідним та вихідним спайками.

- R-STDP (Reward-modulated STDP): Навчання з підкріпленням, де корекція ваг регулюється зовнішнім сигналом винагороди або покарання залежно від правильності діагнозу [25].

- STBP (Spatio-temporal backpropagation): Апаратно-орієнтований алгоритм зворотного поширення помилки, що працює одночасно у просторовій та часовій областях. Він дозволяє проводити прунінг (видалення зайвих зв'язків) та квантування параметрів без значної втрати точності [60].

- Навчання зчитувача (Readout training): Застосовується в резервуарних мережах, де навчаються лише ваги вихідного шару за допомогою лінійної або логістичної регресії, тоді як сам резервуар залишається незмінним [12].

Використання SNN забезпечує значну перевагу в енергоспоживанні порівняно з класичними нейронними мережами (ANN), оскільки обчислення відбуваються лише у відповідь на події (спайки), а «дорогі» операції множення замінюються простим додаванням.

Основні параметри точності та енергоспоживання представлені у табл. 1.3. Із таблиці 1.3 можна зробити висновок, що найбільшої ефективності досягають системи з апаратно-програмним реалізаціями:

- Мінімальні витрати: модель SparrowSNN [61] з нейронами SSF продемонструвала рекордне споживання лише 31.39 нДж на один інференс, що в рази менше за стандартні рішення.

- Оптимізація даних: використання 1-бітного кодування сигналу дозволяє стиснути обсяг даних у 55.9 разів [60].

•Адаптивність: методи синаптичного прунінгу та 6-бітного квантування дозволяють зменшити розмір моделі в 13.6 разів, зберігаючи точність на рівні 98% при споживанні всього 0.75 мкДж на класифікацію [60].

Таблиця 1.3 – Порівняльна таблиця відомих SNN

Джерело	Метрики (точність / чутливість / специфічність)	Енергоспоживання	Кількість класів
<i>Amrita Rana та ін.</i> [59]	85% / - / -	0.176 Вт	2
<i>Haoming Chu та ін.</i> [60]	98.22% / - / -	0.75 мкДж на інференс 0.93 мкВт	5
<i>Dighanchal Banerjee та ін.</i> [29]	94.3% / - / -	-	4
<i>Chia Yee Saw та ін.</i> [12]	95,10% / 97,41% / 93,68%		2
<i>Alireza Amirshahi та ін.</i> [25]	97.9%	1.78 мкДж	2
<i>Zhanglu Yan та ін.</i> [61]	96.72%/ 92,51%/ 75,63% (з детальною інформацією про чутливість та влучність по кожному класу)	31 нДж 6,1 мкВт	4

Таким чином, комбінація спайкінгових обчислень із розрідженим кодуванням робить ці архітектури оптимальними для носимих пристроїв з обмеженим ресурсом елементів живлення.

1.4 Недоліки відомих методів та інформаційних технологій класифікації сигналів ЕКГ та постановка задач дослідження

Аналіз сучасних методів класифікації ЕКГ-сигналів із застосуванням спайкінгових нейронних мереж демонструє, що ці архітектури здатні забезпечувати високу точність класифікації при значно нижчому енергоспоживанні порівняно з традиційними глибокими нейронними мережами та згортковими мережами. Пряма побудова спайкінгової нейронної мережі із використанням оптимізованих правил навчання, таких як STDP із гальмівними механізмами, дозволяє досягати балансу між чутливістю, специфічністю та точністю, що є особливо важливим для застосувань у носимих пристроях із обмеженими ресурсами. Водночас, методи конверсії DNN у SNN забезпечують можливість перенесення вже навчених моделей у енергоефективне середовище, хоча й потребують додаткового перенавчання для компенсації динамічних відмінностей.

Важливим аспектом є вибір способу кодування вхідних сигналів: кодувальник піків (peak encoder) оптимально підходить для компактного подання ключових компонентів серцевого циклу, що сприяє зменшенню обчислювальних витрат, тоді як кодувальник у Гаусові рецептивні поля краще відображає плавні зміни хвиль. Комбіноване застосування різних методів кодування та попередньої обробки у мультиканальних архітектурах відкриває нові можливості для підвищення продуктивності моделей без значного збільшення енергоспоживання.

Проблеми масштабованості моделей залишаються викликом, особливо при збільшенні кількості класів та складності вхідних даних. Для SNN характерне зростання щільності спайків, що може призводити до перенасичення обчислювальних блоків і зниження розпізнавальної здатності. Вирішення цих питань вимагає впровадження адаптивних механізмів контролю активності, багаторівневого навчання та модульного розширення архітектури. Перспективи розвитку нейроморфних обчислень у класифікації

ЕКГ пов'язані з інтеграцією покращених правил навчання, таких як поєднання STDP із модуляцією винагороди, а також із впровадженням механізмів уваги для більш гнучкого реагування на різні фази серцевого циклу.

Використання спеціалізованих нейроморфних процесорів дозволяє реалізувати повний цикл аналізу на рівні сенсорів, що знижує потребу у передачі великих обсягів даних і підвищує автономність систем. Загалом, розвиток SNN у поєднанні з оптимізованими методами кодування та навчання відкриває шлях до створення ефективних, енергоощадних і точних систем для безперервного моніторингу серцевої діяльності. Це особливо актуально для носимих пристроїв, де поєднання продуктивності та низького енергоспоживання є ключовим фактором успіху. Подальше дослідження має зосереджуватися на подоланні викликів масштабованості, підвищенні адаптивності моделей до різноманітних клінічних сценаріїв та інтеграції нейроморфних рішень у комплексні медичні системи підтримки прийняття рішень.

Дисертаційна робота присвячена побудові інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі для виявлення аномальних ударів серця, яка орієнтована на використання у носимих пристроях.

1.5 Висновки до першого розділу

У розділі показано, що задача побудови інформаційних технологій автоматизованої класифікації сигналів ЕКГ є актуальною через потребу оперативного неперервного діагностування наявності аритмії серцебиття. Сучасні виклики вимагають побудови таких класифікаторів ЕКГ, які можна було б використовувати у носимих приладах для оперативного та дистанційного автоматизованого моніторингу та телемедицини. А для цього вони повинні бути ресурсоефективними (споживати мінімум енергії і потребувати мінімум обчислювальних ресурсів та пам'яті), а їх швидкодія

повинна бути достатньою для функціонування в реальному часі, тобто з мінімальною затримкою. Розглянуто методи класифікації сигналів ЕКГ та висвітлено, що методи на основі машинного навчання та глибоких нейронних мереж характеризуються високою точністю класифікації ЕКГ, але мають велике споживання енергії та потребують великих обчислювальних ресурсів та пам'яті. Перспективними є методи на основі спайкінгових нейронних мереж, які є енерго- та ресурсо-ефективними, працюють у реальному часі і мають не набагато гіршу точність, ніж методи на основі глибоких нейронних мереж.

Розглянуто методи класифікації сигналів ЕКГ на основі спайкінгових нейронних мереж та зроблено висновок, що вони не завжди використовують оптимальну структуру мережі, методи навчання спайкінгових нейронних мереж ще не є досконалими, а методи попереднього перетворення та кодування сигналів ЕКГ для подання у спайкінгову нейронну мережу часто призводять до втрати корисної інформації, а тому потребують вдосконалення. Усунення зазначених недоліків дозволить підвищити точність роботи ресурсоефективних класифікаторів ЕКГ, що і є метою даного дисертаційного дослідження.

Основні результати, викладені у розділі, опубліковано в роботах [1], [2], [6], [7].

РОЗДІЛ 2

МОДЕЛІ ТА МЕТОДИ ДЛЯ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ НА ОСНОВІ СПАЙКІНГОВИХ НЕЙРОННИХ МЕРЕЖ

2.1 Моделі спайкінгових нейронів, що використовуються при класифікації сигналів ЕКГ

Моделі спайкінгових нейронів відрізняються за рівнем біологічної достовірності, обчислювальної складності та можливостями адаптації до специфічних задач. У контексті класифікації сигналів ЕКГ, вибір моделі впливає на точність і швидкість аналізу. LIF (Leaky Integrate-and-Fire) демонструє збалансоване співвідношення між простотою реалізації та здатністю до адекватного відтворення підпорогової динаміки мембранного потенціалу [63], [64].

На відміну від моделі Ходжкина-Хакслі [65], яка описує мембранний потенціал через систему нелінійних диференціальних рівнянь з урахуванням механіки іонних каналів, LIF абстрагує ці фізіологічні деталі. Це дозволяє уникати високих обчислювальних витрат на моделювання кожної часової фази потенціалу дії та працювати з великими мережами при помірному споживанні ресурсів [66]. Модель Ходжкина-Хакслі переважає в точності відтворення внутрішніх мембранних процесів, проте потребує надзвичайно малих кроків інтеграції для фіксації крутих профілів сплесків, що уповільнює симуляцію масштабних систем [65].

Spike Response Model (SRM), своєю чергою, пропонує вищий рівень абстракції порівняно з LIF і формує мембранний потенціал із сум ефектів попередніх вхідних та власних вихідних спайків через згорткові функції ядра [63]. Це забезпечує гнучке налаштування форми синаптичного внеску та післяспайкової реакції, корисне для задач із виразними часовими кореляціями сигналів. Проте SRM потребує вибору й оптимізації форми ядра під конкретні дані, що може ускладнювати реалізацію при роботі у режимах реального часу

[67]. LIF-модель має перевагу завдяки математичній простоті: її поведінку можна описати лінійним диференціальним рівнянням для мембранного потенціалу з термінами витоку та інтеграції вхідного струму [64]. Це робить її сумісною з ефективними чисельними методами і забезпечує стабільну роботу навіть при великій кількості нейронів та довготривалому навчанні.

Таким чином, порівнюючи поведінку та характеристики моделей Ходжкина-Хакслі, SRM і LIF при аналізі ЕКГ-сигналів зроблено висновок, що LIF пропонує оптимальний баланс між обчислювальною ефективністю й достатньою здатністю відтворення ключових аспектів динаміки мембрани для цілей класифікації [63], [66]. Її варіативність дозволяє розширювати можливості без радикальної зміни базового принципу роботи.

Побудова математичного опису моделі Leaky Integrate-and-Fire (LIF) базується на електричній аналогії нейрона із RC-колом, де мембрана розглядається як комбінація ємності C_m та резистивного витоку з опором R [68]. Вхідні імпульси від попереднього шару спайкінгових нейронів перетворюються на синаптичний струм $I(t)$, який заряджає мембрану. Відповідно до другого закону Кірхгофа, рівняння зміни мембранного потенціалу $U(t)$ має вигляд:

$$C_m \frac{dU(t)}{dt} = -\frac{U(t) - U_{\text{rest}}}{R} + I(t) \quad (2.1)$$

де термін $-\frac{U(t)-U_{\text{rest}}}{R}$ описує витік заряду через резистивний компонент мембрани при відсутності входу. Переписавши опір R через сталу часу мембрани $\tau_m = RC_m$, отримаємо диференціальне рівняння першого порядку:

$$\tau_m \frac{dU(t)}{dt} = -[U(t) - U_{\text{rest}}] + R I(t) \quad (2.2)$$

Ця форма є основою моделі LIF [69], що описує плавну зміну потенціалу до моменту активації спайка. Коли $U(t)$ досягає порогового значення $U_{threshold}$, відбувається генерація спайка у дискретному представленні:

$$S(t) = \Theta(U(t) - U_{threshold}) \quad (2.3)$$

де $\Theta()$ – функція Хевісайда [70]. Після активації спайка потенціал миттєво скидається до U_{reset} , що моделює фазу рефрактерності та запобігає негайній повторній активації. Аналіз показує, що вирішення цього рівняння дає експонентно зростаючу або спадну поведінку $U(t)$ залежно від знаку вхідного стимулу. Для постійного струму I_0 , розв’язок у формі початкової задачі:

$$U(t) = U_{rest} + [U(0) - U_{rest}]e^{-\frac{t}{\tau_m}} + RI_0 \left(1 - e^{-\frac{t}{\tau_m}}\right) \quad (2.4)$$

Ця залежність демонструє інтегруючу природу моделі та ефект витоку, навіть при тривалому вході потенціал асимптотично прямує до певного стаціонарного рівня, але не зростає безкінечно. Практичне застосування моделі LIF у SNN виправдане тим, що вона підтримує часові властивості нейрона та одночасно уникає складної кінетики каналів моделі Ходжкина-Хакслі [65]. Це робить модель LIF оптимальною для апаратних реалізацій та масштабних симуляцій.

Для візуалізації динаміки наведено графік мембранного потенціалу з витоком і спайками (рис. 2.1). На графіку показано, як вхідний струм накопичується, викликаючи спайки, а витік забезпечує повернення до рівноваги. Така модель ефективна для ЕКГ, де сигнали мають імпульсний характер.

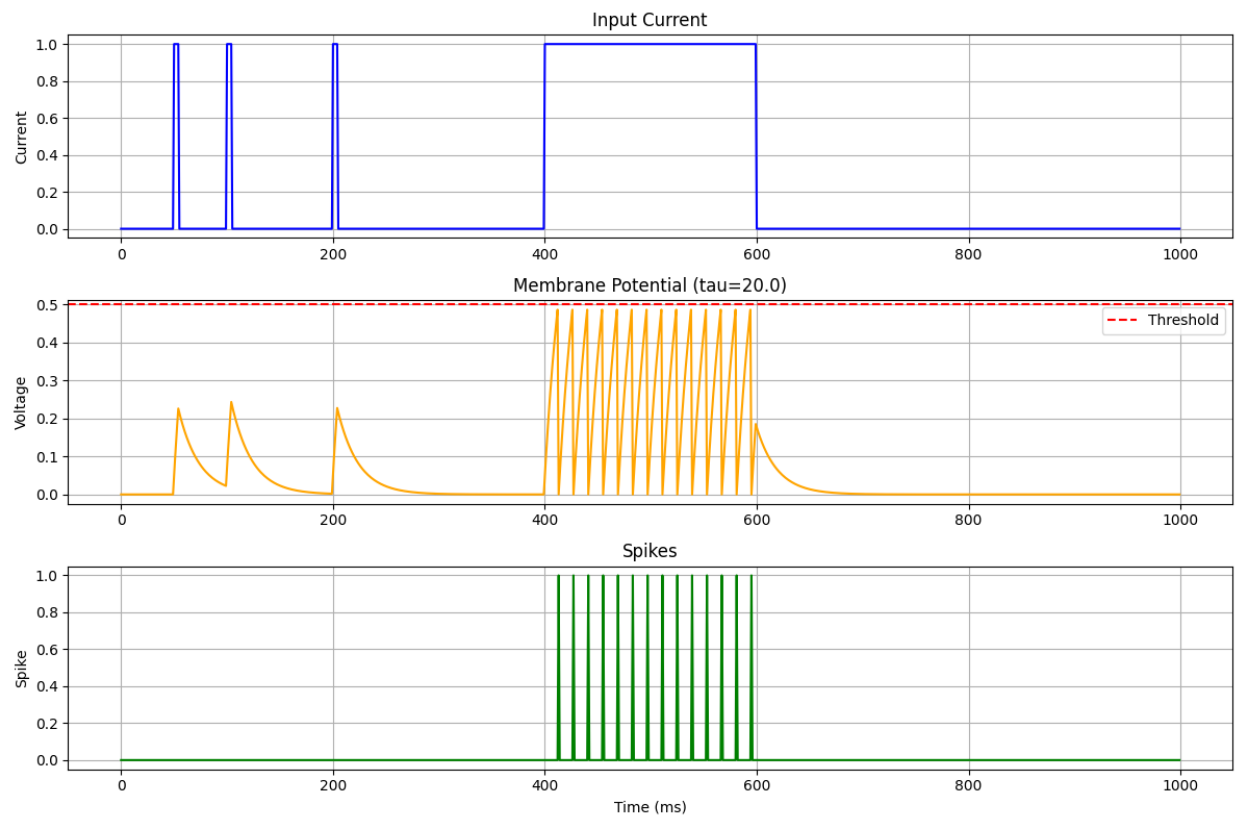


Рисунок 2.1 – Приклад генерації спайків та динаміки мембранного потенціалу

Параметри моделі LIF адаптовані для різних шарів мережі з метою оптимізації обробки ЕКГ-сигналів. У вхідному шарі постійна часу $\tau_{\text{input}} = 15.0$ мс, що забезпечує швидку реакцію на динамічні зміни сигналу, такі як швидкі пікові амплітуди в ЕКГ (наприклад, R-хвилі). Це дозволяє нейронам швидко інтегрувати короточасні імпульси, мінімізуючи затримки в кодуванні [71].

У резервуарі застосовується $\tau_{\text{ism}} = 40.0$ мс, що створює більш повільну динаміку для збереження інформації в часі. Така конфігурація формує згасаючу пам'ять, де попередні стани впливають на поточні, що критично для аналізу часових шаблонів у ЕКГ, таких як інтервали RR або аритмічні послідовності. Приклад графіку згасання мембранного потенціалу для різних τ наведено на рис. 2.2.

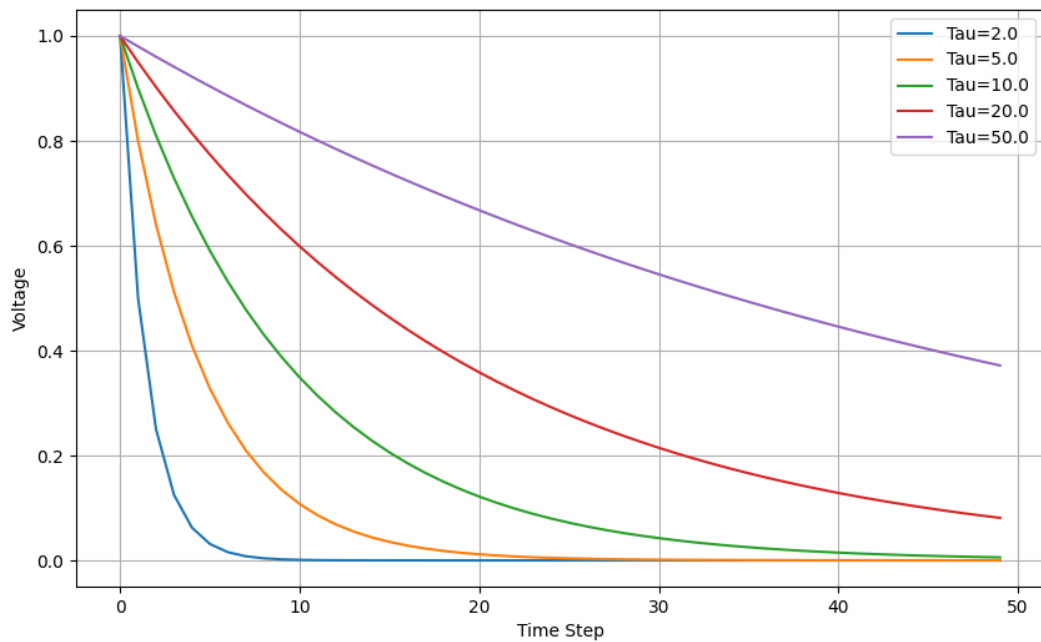


Рисунок 2.2 – Графік згасання мембранного потенціалу для різних τ

Вибір моделі LIF для моделювання нейронних елементів у задачах класифікації часових рядів, таких як ЕКГ, ґрунтується на кількох практичних і теоретичних перевагах цієї моделі над більш деталізованими біофізичними підходами. Модель Ходжкіна-Хакслі, хоча і забезпечує максимально наближену до реальних процесів картину поведінки мембрани завдяки опису кінетики відкривання та закривання іонних каналів [72], потребує інтеграції системи нелінійних диференціальних рівнянь високого порядку для кожного елемента мережі. Це означає різке збільшення обчислювальної складності при масштабуванні до тисяч нейронів, особливо у режимі симуляції спільної дії всіх компонентів. LIF натомість зберігає основну фізичну аналогію з RC-колом та моделює процес накопичення заряду й витоку через єдину змінну, мембранний потенціал, що робить її легкою для чисельної реалізації. Перевага LIF полягає у вдалому балансі між біологічною правдоподібністю й ефективністю розрахунків [73]. Вона враховує ключові часопросторові характеристики відповіді нейрона, інтегрування вхідного сигналу і його витік без надмірної деталізації мікропроцесів на рівні окремих каналів. Це особливо корисно при роботі з часовими рядами, де важливо утримувати інформацію

про шаблони імпульсів протягом кількох мілісекунд або навіть секунд залежно від τ . Гнучкість налаштування сталої часу мембрани дає можливість адаптувати нейрон до різних масштабів часової структури вхідних даних.

2.2 Архітектура спайкінгової нейронної мережі

Для спайкінгової нейронної мережі перевагою використання LSM (Liquid State Machine, Машина рідинного стану) та резервуарного числення є здатність перетворювати просторово-часові шаблони сигналу у багатовимірні представлення, які придатні для подальшої класифікації. Механізм роботи резервуарного шару ґрунтується на рекурентному з'єднанні великої кількості нейронів із випадково визначеною топологією та фіксованими вагами, що створює високорозмірну динаміку у відповідь на поточний вхідний сигнал [74]. Вхідний сигнал проектується у внутрішній стан мережі, де його компоненти, як у просторі, так і у часі, взаємодіють відповідно до властивостей збуджувальних та гальмівних зв'язків [75]. Це дає змогу кодувати інформацію про послідовність подій навіть після завершення окремого етапу вхідного впливу. У класичній інтерпретації резервуарного підходу, до якого відноситься LSM, низьковимірні сигнали нелінійно відображаються у високовимірний простір станів, що дозволяє перетворювати задачі на лінійно роздільні. Така проекція особливо цінна для обробки ЕКГ, оскільки серцеві скорочення та їх аномалії проявляються як складні часові структури з поєднанням коротких і довгих залежностей між компонентами сигналу. Рекурентні зв'язки всередині резервуарного шару створюють ефект згасаючої пам'яті, при якому інформація про стимул зберігається деякий час після його завершення. Завдяки цьому можлива інтеграція відомостей про кілька стимулів послідовно, що забезпечує захоплення часових закономірностей і підтримку контексту для класифікаційного шару. Структура резервуарних зв'язків може мати додатково впорядковані властивості. Це дозволяє покращити інтеграцію просторових і часових характеристик сигналу

без втрати здатності реагувати на локальні зміни. Додатково така структура сприяє збереженню властивості резервуару, тобто розвитку динаміки моделі без залежності від початкових умов, та забезпечує ефективне поширення сигналів усередині мережі [76]. Таке структурне налаштування підсилює обробку комплексних ЕКГ шаблонів і формує стабільну реакцію LSM у шумових середовищах. Важливо й те, що різновиди зв'язків у резервуарі, збуджувальні й гальмівні, виконують доповнювані функції. Збуджувальні зв'язки поширюють і підсилюють активність серед нейронів шару, тоді як гальмівні взаємодії регулюють рівень збудження та запобігають надмірній синхронізації чи хаотичним коливанням потенціалу [75]. Ще одне підтвердження придатності LSM для роботи зі складними просторово-часовими завданнями міститься у результатах обчислювальних моделей, де затримки в рекурентних зв'язках дозволяють генерувати комплексну нелінійну динаміку та ефективно класифікувати вхідні патерни [76]. Таким чином LSM виконує не тільки більшість стандартних завдань рекурентної обробки для ЕКГ сигналів, вона також додає характеристику високовимірної проекції вхідного сигналу. Через поєднання структурної організації й фізичних параметрів зв'язків цей підхід демонструє адаптивність до вимог конкретної задачі медичної діагностики й підтримує баланс між стабільністю й чутливістю до змін сигналу. Використання таких архітектур дозволяє апаратним реалізаціям отримувати стабільний результат при роботі в режимах реального часу за рахунок природного кодування просторово-часових залежностей всередині резервуарного шару [74], [76].

Архітектура мережі для класифікації сигналів ЕКГ, розроблена у цій роботі, зображена на рис. 2.3.

Вхідний шар приймає кодовані значення сигналу ЕКГ та перетворює цю інформацію у спайки, які в подальшому обробляються в резервуарі. У даному випадку було використано кодування Гаусовими рецептивними полями (Gaussian Receptive Fields, GRF), яке більш детально описано у п. 2.4. Це перетворення забезпечує одночасно високу точність відтворення форми хвиль

та стійкість до варіацій амплітуди, що особливо важливо при аналізі складних біометричних серій. Конфігурація вхідного шару передбачає 50 незалежних LIF-нейронів, кожен з яких асоційований із власним рецептивним полем, центрованим навколо μ_i – фіксованого значення.

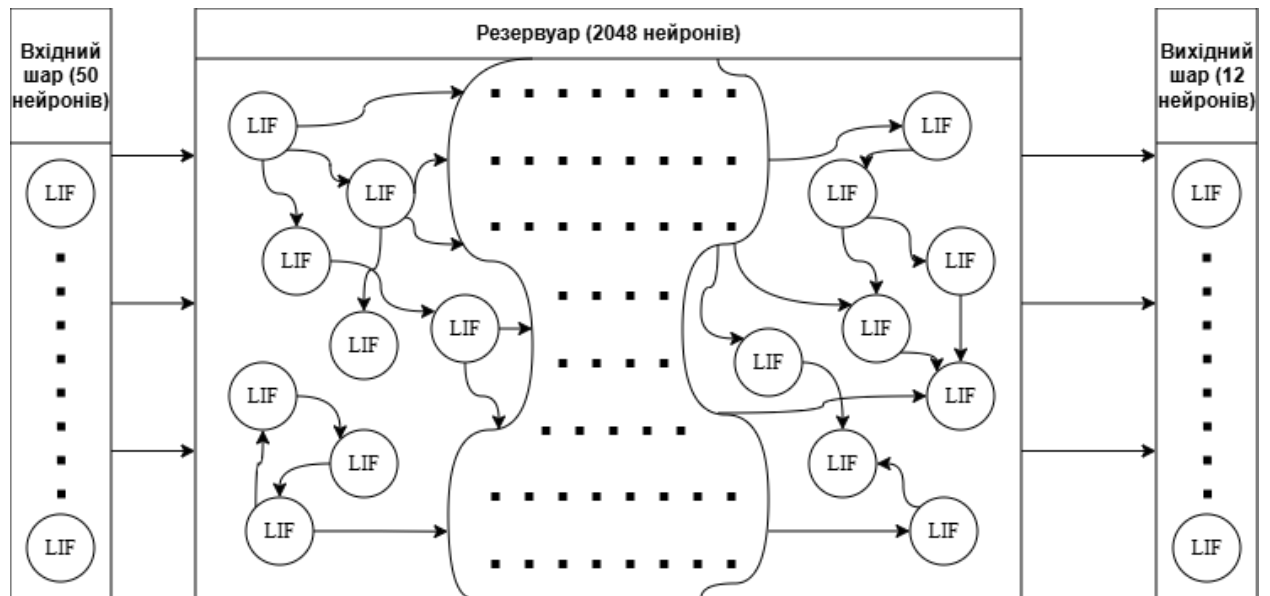


Рисунок 2.3 – Архітектура SNN для класифікації сигналів ЕКГ

Резервуарний шар реалізовано у вигляді рекурентної мережі типу Liquid State Machine (LSM), в якій присутні два типи зв'язків між нейронами: збуджувальні (E) та гальмівні (I). Співвідношення між ними відповідає пропорції 4/1, що спостерігається в сенсорній корі ссавців. Така вибірка підтримує як інтенсивну генерацію і поширення імпульсів у межах шару, так і стабілізацію активності через гальмівні впливи. У резервуарі відсутнє розподілення нейронів на збуджувальні та гальмівні – за це відповідають синапси між нейронами. Вхідний шар підключено до резервуара за допомогою розріджених зв'язків, де всі зв'язки збуджувальні.

У самому резервуарі міститься 2048 нейронів, організованих зі щільністю випадкових з'єднань близько 10%. Це означає, що кожен нейрон підключений лише до обмеженої кількості інших вузлів, а матриця ваг W_{res} є розрідженою. Така щільність контролює баланс між інформативністю стану резервуару та кількості обчислень синаптичних подій, проте надмірна

кількість зв'язків може призвести до того, що весь резервуар реагуватиме однаково на різні входи, втрачаючи ефект розділення траєкторій [77]. Матриця з'єднань будується з дотриманням закону Дейла: фіксована пропорція збуджувальних та гальмівних елементів (80% збуджувальних і 20% гальмівних). Формально це можна описати так: для набору індексів $\mathcal{E}$ (збуджувальні) $w_{ij} > 0$, а для $\mathcal{I}$ (гальмівні) $w_{ij} < 0$, де w_{ij} – вага синаптичного зв'язку від нейрона i до нейрона j . Абсолютне значення ваг початково вибирається з невеликого діапазону але не виходить за межі $[-1; 1]$. Подібні механізми забезпечують реалістичне моделювання короткочасних змін ефективності синапсу та формують внутрішню часову пам'ять шару.

Вихідний шар описаної архітектури складається з нейронів, кожен з яких відповідає окремій категорії класифікації електрокардіографічних сигналів. Кількість нейронів на виході визначається числом цільових класів, і для задачі класифікації сигналів ЕКГ використовується 12 вихідних нейронів. Резервуар з'єднаний з вихідним шаром із розрідженими зв'язками величиною 20%, і розподілення цих зв'язків на збуджувальні та гальмівні зберігає пропорцію резервуару (80% збуджувальних і 20% гальмівних).

Вихідний шар отримує сигнали у вигляді просторово-часового паттерну імпульсів (спайків) нейронів резервуару (так званий «рідкий стан» резервуару) протягом всього циклу обробки вхідної спайкінгової послідовності. Кожен нейрон вихідного шару є «індикатором» певного «рідкого стану» (в процесі навчання налаштовується реагувати високою частотою вихідних імпульсів на певний «рідкий стан»). Прогнозований клас визначається тим вихідним нейроном, який видає максимальну кількість спайків серед усіх інших вихідних вузлів за тривалість одного періоду вхідного сигналу ЕКГ. Чим більша кількість вихідних спайків конкретного вихідного нейрона відносно інших вихідних нейронів, тим більша впевненість у вірній класифікації..

2.3 Розробка методу навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ

У сучасній науковій літературі описано значну кількість методів навчання рекурентних та спайкінгових нейронних мереж, зокрема алгоритм зворотного поширення помилки крізь час (Backpropagation Through Time, BPTT) [78], [79], рекурентне навчання в реальному часі (Real-Time Recurrent Learning, RTRL) [80], [81], методи лінійної регресії для вихідного шару в архітектурах з резервуарним обчисленням, а також гібридні підходи на основі FORCE-навчання [82], [83] та еволюційних алгоритмів [84]. Ці методи широко застосовуються для моделювання динамічних процесів і часових послідовностей, проте демонструють низку суттєвих обмежень. Зокрема, класичний BPTT страждає від проблем зникаючих та вибухових градієнтів, що призводить до нестабільності навчання. Аналогічні недоліки притаманні й іншим градієнтним методам, тоді як методи без учителя або з частковим навчанням часто не забезпечують достатньої точності для складних нелінійних задач. Унаслідок цього проблема побудови ефективних, стабільних та масштабованих методів навчання для СНМ, особливо з урахуванням нейронів резервуару, продовжує залишатися відкритою науковою задачею сучасних досліджень у сфері нейроморфних обчислень та штучного інтелекту.

Для розв'язання наукової задачі даного дисертаційного дослідження за основу взято алгоритм зворотного поширення помилки крізь час, який доповнено низкою спеціальних процедур, спрямованих на оптимізацію початкової конфігурації ваг, підвищення стабільності градієнтного потоку та покращення загальної ефективності моделі. Такий гібридний підхід дозволяє поєднати переваги традиційного BPTT із сучасними техніками ініціалізації та тонкого налаштування

Розроблений метод навчання складається з трьох послідовних етапів, кожен з яких виконує чітко визначену функцію в загальному процесі оптимізації:

1) Етап кластерної ініціалізації ваг вихідних нейронів. На цьому попередньому етапі здійснюється визначення оптимальної структури зв'язків та раціональна ініціалізація вагових коефіцієнтів між нейронами резервуару і нейронами вихідного шару. Такий підхід, на відміну від стандартної випадкової ініціалізації, забезпечує більш структуровану початкову конфігурацію, зменшує ризик потрапляння в локальні мінімуми та сприяє кращій адаптації мережі до особливостей вхідних даних.

2) Основний етап керованого навчання за допомогою алгоритму ВРТТ. На даному етапі реалізується класичне зворотне поширення помилки крізь час з урахуванням розгорнутої в часі структури мережі, що дозволяє ефективно налаштовувати параметри резервуару та вихідних з'єднань на основі градієнтного спуску.

3) Етап керованого помилкою точного налаштування вихідного зчитувального шару. Завершальний етап передбачає додаткову оптимізацію параметрів вихідного шару мережі з використанням адаптивних технік, що забезпечує подальше підвищення точності, стабільності та узагальнювальної здатності моделі, а також мінімізацію залишкових ефектів зникаючих градієнтів.

Запропонований 3-етапний метод представляє собою комплексне рішення, яке поєднує переваги існуючих підходів і водночас усуває їхні ключові недоліки, що робить його перспективним для практичного застосування в задачах моделювання складних динамічних систем на базі СНМ. Подальші експериментальні дослідження підтверджують його ефективність порівняно з базовими методами.

Узагальнену схему методу навчання наведено на рис. 2.4.

Перший етап методу навчання спрямований на корекцію вихідного шару машини рідинних станів (LSM) на основі кореляційного аналізу, є етапом ініціалізації та оптимізації вагових зв'язків між резервуаром і вихідним шаром. Ідея методу полягає у визначенні «найкращих» нейронів резервуару для ініціалізації збуджувальних та гальмівних зв'язків з нейронами вихідного

шару, базуючись на їх активності залежно від класу вхідних даних. Цей метод застосовується після початкової ініціалізації мережі та симуляції її роботи на тренувальних даних, коли резервуарний шар уже генерує динамічні стани (рідинні стани) у відповідь на вхідні сигнали. Основна мета – встановити зв'язки нейронів вихідного шару та їхні ваги таким чином, щоб вони відображали статистичну кореляцію між активністю нейронів резервуару та цільовими класами, без використання градієнтного спуску на цьому етапі. Це дозволяє швидко адаптувати вихідний шар до шаблонів активності, забезпечуючи базову класифікацію, особливо в умовах незбалансованих даних, як у задачах класифікації сигналів ЕКГ.

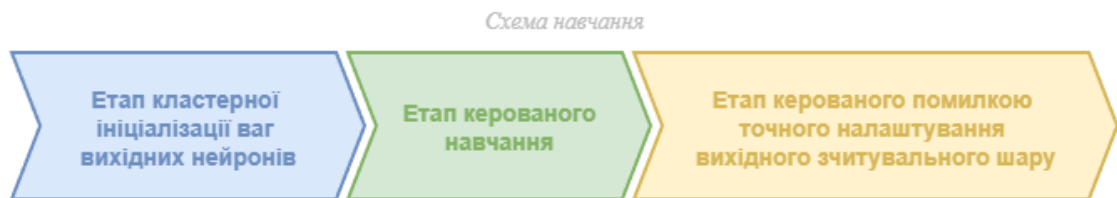


Рисунок 2.4 – Узагальнена схема методу навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ

Процес починається з переведення моделі в режим оцінювання, щоб уникнути оновлення параметрів під час проходження даних. Далі відбувається збір статистичних даних про активність нейронів резервуару: для кожного зразка в тренувальному наборі даних модель симулює динаміку мережі, фіксуючи спайки (спайкінгові події) в резервуарному шарі протягом усього часового ряду вхідного сигналу. Ці спайки сумуються по часу для кожного зразка, утворюючи вектор середньої активності резервуарних нейронів для відповідного зразка. Потім ці вектори агрегуються за класами: для кожного класу обчислюється сумарна активність кожного нейрона резервуару по всіх зразках цього класу, а також загальна кількість зразків у класі. Це дозволяє розрахувати середню активність (середню частоту спайків) для кожного нейрона резервуару по кожному класу.

Наступним кроком є нормалізація цих середніх значень за допомогою Z-оцінки, яка перетворює сирі значення активності на відхилення від середнього, нормалізовані на стандартне відхилення. Для цього спочатку обчислюється середня активність по всіх класах для кожного нейрона (середнє по класах), а також стандартне відхилення по класах. Щоб уникнути поділу на нуль або ампліфікації шуму в нейронах з низькою активністю, до стандартного відхилення додається невелика константа (шумовий поріг), пропорційна середній активності (наприклад, 5% від середнього плюс мінімальне значення для стабільності). Отримані Z-оцінки інтерпретуються як міра кореляції: позитивні значення вказують на активацію нейрона для певного класу (збуджувальний вплив), негативні – на гальмування (гальмівний вплив). Це забезпечує вирівнювання внеску "голосних" (високоактивних) і "тихих" (низькоактивних) нейронів, акцентуючи на специфічних патернах активації для кожного класу.

Далі застосовуються структурні обмеження для забезпечення біологічної правдоподібності та ефективності мережі: щільність (sparsity) зв'язків на рівні 20% (тобто лише 20% нейронів резервуару з'єднуються з кожним вихідним нейроном) і співвідношення збуджувальних/гальмівних зв'язків (наприклад, 80% збуджувальних і 20% гальмівних). Для кожного класу окремо з вектора кореляцій (Z-оцінок) виділяються позитивні значення (кандидати на збуджувальні ваги) і негативні значення (кандидати на гальмівні ваги). З позитивних значень відбираються топ-N збуджувальних (де N – 80% від загальної кількості з'єднань, наприклад, 160 для резервуару з 2048 нейронів), враховуючи лише ті, що перевищують нуль. Аналогічно, з негативних відбираються топ-M гальмівних (де M – 20%, наприклад, 40), фокусуючись на найбільш негативних значеннях. Якщо доступних кандидатів менше, ніж потрібно, використовуються всі доступні.

Відібрані значення нормалізуються: для збуджувальних – діляться на максимальне позитивне значення в групі, приводячи до діапазону $[0, 1]$; для гальмівних – діляться на абсолютне значення мінімального негативного,

приводячи до діапазону $[-1, 0]$. Це забезпечує уніфікацію шкали ваг. Отримані ваги присвоюються матриці вихідного шару, формуючи розріджену матрицю зі змішаними знаками.

Цей метод є ефективним для швидкої адаптації вихідного шару, оскільки не вимагає ітеративних обчислень градієнтів, а покладається на статистичний аналіз активності. Він сприяє покращенню чутливості для менш представлених класів, зменшуючи перехресну кореляцію між класами, і готує мережу до подальшої тонкої настройки. У контексті дисертації цей етап підкреслює біонатхненний підхід до навчання, імітуючи синаптичну пластичність на основі кореляцій активності, подібно до навчання Гебба [85], [86], але з контрольованими обмеженнями для стабільності та ефективності в задачах класифікації часових рядів, таких як аналіз ЕКГ-сигналів.

Покроковий алгоритм 1-го етапу:

- 1) Перевести модель у режим оцінювання для фіксації параметрів.
- 2) Ініціалізувати матриці накопичення: матрицю сум спайків резервуару по класах (розмір: кількість класів на кількість нейронів резервуару) і вектор кількості зразків по класах.
- 3) Для кожного батчу в тренувальному завантажувачі:
 - Виконати прохід моделі з фіксацією активності (спайків резервуару по часу).
 - Сумувати спайки по часу для кожного зразка, отримуючи вектори активності.
 - Для кожного зразка в батчі: додати вектор активності до сумарного для відповідного класу і інкрементувати лічильник зразків класу.
- 4) Обчислити середню активність по класах: розділити сумарні спайки на кількість зразків для кожного класу.
- 5) Обчислити Z-оцінки: розрахувати середнє і стандартне відхилення по класах для кожного нейрона.
- 6) Для кожного класу:

- Виділити позитивні та негативні кореляції.
- Відібрати топ збуджувальних (позитивних) і топ гальмівних (негативних) за кількістю, визначеною щільністю зв'язків та співвідношенням.
- Нормалізувати відібрані значення до $[0,1]$ для збуджувальних і $[-1; 0]$ для гальмівних.
- Сформувати матрицю ваг для класу.

7) Присвоїти отриману матрицю ваг вихідному шару.

8) Зберегти модель.

Схему алгоритму першого етапу навчання наведено на рис. 2.5.

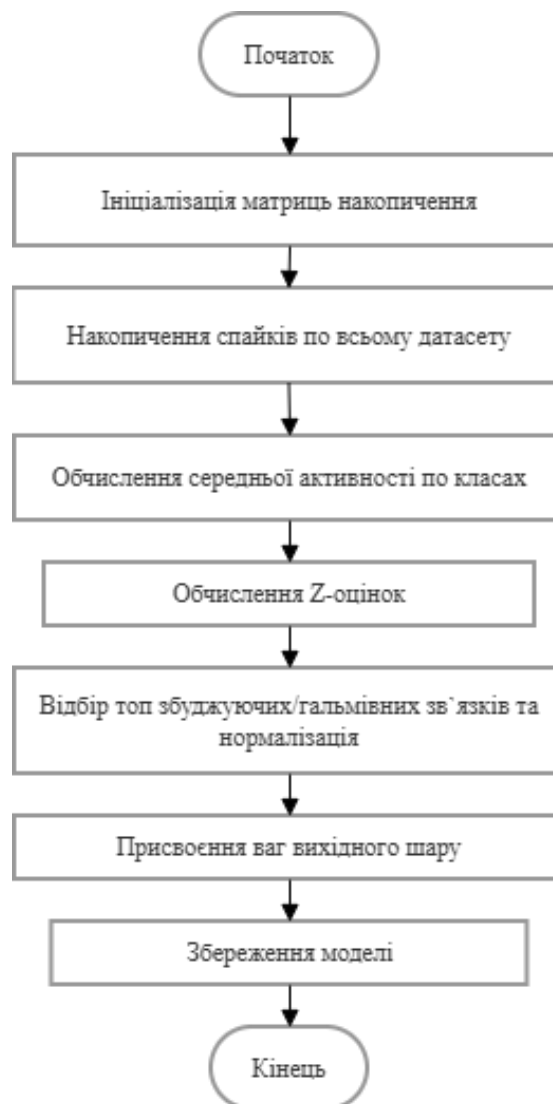


Рисунок 2.5 – Схема алгоритму першого етапу навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ

Другий етап методу навчання, заснований на градієнтному спуску, є основним етапом оптимізації всієї мережі, включаючи вхідний шар, резервуарний шар і вихідний шар. Він застосовується після попередньої кореляційної ініціалізації або на новій моделі, щоб досягти високої точності класифікації шляхом повного налаштування параметрів. Цей етап використовує відомий підхід керованого навчання для спайкінгових нейронних мереж, адаптований до динаміки LSM, з урахуванням часового аспекту сигналів і необхідності стабілізації градієнтів. Мета – мінімізувати помилку класифікації на тренувальних даних, покращуючи узагальнення на тестових, особливо в умовах варіабельної довжини вхідних послідовностей, як у медичних часових рядах. Для цього етапу був використаний алгоритм зворотного поширення помилки крізь час, проте може бути використаний будь-який алгоритм навчання з учителем.

Алгоритм зворотного поширення помилки крізь час (Backpropagation Through Time, BPTT) [81] є одним із фундаментальних методів навчання рекурентних нейронних мереж, що дає змогу ефективно враховувати часову залежність у послідовних даних. Запропонований П. Вербосом [80] наприкінці 1980-х років, цей алгоритм являє собою природну адаптацію класичного алгоритму зворотного поширення помилки до динамічних систем зі зворотними зв'язками в часі. BPTT дозволяє перетворити рекурентну архітектуру на еквівалентну глибоку мережу прямого поширення, що робить можливим застосування градієнтних методів оптимізації для оновлення параметрів моделі з урахуванням усієї часової історії.

Сутність методу полягає в розгортанні рекурентної нейронної мережі на задану кількість часових кроків T . Кожний часовий крок представляє собою копію оригінальної мережі, причому вагові матриці залишаються спільними для всіх зрізів. Рекурентні зв'язки таким чином перетворюються на прямі зв'язки між послідовними шарами розгорнутої структури. Завдяки цьому обчислення градієнтів функції втрат відносно параметрів здійснюється за

допомогою ланцюгового правила диференціювання, що поширюється назад через усі часові кроки послідовності.

Навчання відбувається по епохах: для кожної епохи модель переведена в режим тренування, і дані проходять через ітератор завантажувача. Для кожного батчу вхідні дані адаптуються до фіксованої форми (доповнюються нулями для змінної довжини, додається вимір для каналів), переміщуються на пристрій обчислень і перетворюються на тензори з плаваючою комою. Цільові мітки також переміщуються на пристрій. Оптимізатор очищає градієнти, після чого проводиться прямий прохід моделі, обчислюючи вихідні значення нейронної мережі (логіти), що є частотами спайків у вихідному шарі, на основі середньої частоти спайків вихідного шару. Втрата обчислюється між логітами та мітками, після чого проводиться зворотний прохід для розрахунку градієнтів.

Щоб уникнути вибуху градієнтів, характерного для рекурентних мереж і SNN з часовими залежностями, застосовується обмеження градієнтів з нормою 1.0, обмежуючи їхню величину. Оптимізатор виконує крок оновлення параметрів. Під час епохи відстежуються накопичувальні втрати, точність і проміжні метрики, з виводом кожні 100 кроків для моніторингу. Після завершення епохи планувальник оновлює швидкість навчання, і обчислюються середні втрати та точність за епоху.

Після кожної епохи проводиться оцінювання на тестовому наборі: модель переведена в режим оцінювання, обчислюється чутливість (середня по класах), і якщо ця метрика краща за попередній найкращий, модель зберігається. Це забезпечує вибір найкращої ітерації за ключовою метрикою. 2й етап завершується після заданої кількості епох, з виводом найкращої чутливості.

У дисертаційному контексті цей метод демонструє інтеграцію традиційного глибокого навчання з СНМ, дозволяючи проводити повну оптимізацію, що покращує відокремленість рідинних станів і адаптацію до складних патернів. Він враховує специфіку спайкінгових мереж, і його

ефективність підтверджується метриками, такими як чутливість, влучність і F1-score, роблячи мережу придатною для реального застосування в кардіодіагностиці.

Покроковий алгоритм:

- 1) Ініціалізувати функцію втрат, оптимізатор і планувальник.
- 2) Для кожної епохи:
 - Перевести модель у режим тренування.
 - Для кожного батчу в тренувальному завантажувачі:
 - Подати вхідні дані
 - Виконати прямий прохід.
 - Обчислити функцію втрат.
 - Виконати зворотний прохід для градієнтів.
 - Застосувати обмеження градієнтів (норма 1.0).
 - Виконати крок оптимізатора.
 - Оновити накопичувальні метрики (втрати, точність).
 - Оновити швидкість навчання планувальником.
 - Обчислити і вивести середні втрати та точність за епоху.
 - Виконати оцінювання на тестовому наборі: обчислити чутливість.
 - Якщо чутливість краща, зберегти модель як найкращу.
- 3) Після всіх епох вивести найкращу чутливість.

Схему алгоритму другого етапу навчання наведено на рис. 2.6.

Третій етап методу навчання. Задля досягнення максимальної точності навчання було також включено фінальний етап тренування мережі. Цей етап тренування є спеціалізованою процедурою тонкого налаштування (fine-tuning) ваг вихідного шару мережі, яка застосовується після попередніх етапів кореляційного аналізу та градієнтного спуску. Цей етап фокусується на ітеративній корекції ваг на основі аналізу помилок класифікації для кожного окремого зразка даних, з акцентом на посилення специфічних патернів активації. На відміну від глобальних методів, таких як градієнтний спуск, цей підхід використовує правило-орієнтоване оновлення, натхненне біологічними

механізмами синаптичної пластичності (наприклад, подібно до STDP – Spike-Timing-Dependent Plasticity, але адаптоване до частотного кодування у вихідному шарі). Він спрямований на виправлення випадків, коли модель неправильно класифікує зразок, посилюючи збуджувальні (позитивні) ваги для правильного класу та посилюючи гальмівні (негативні) ваги для неправильних класів, щоб зменшити ймовірність помилкових активацій.

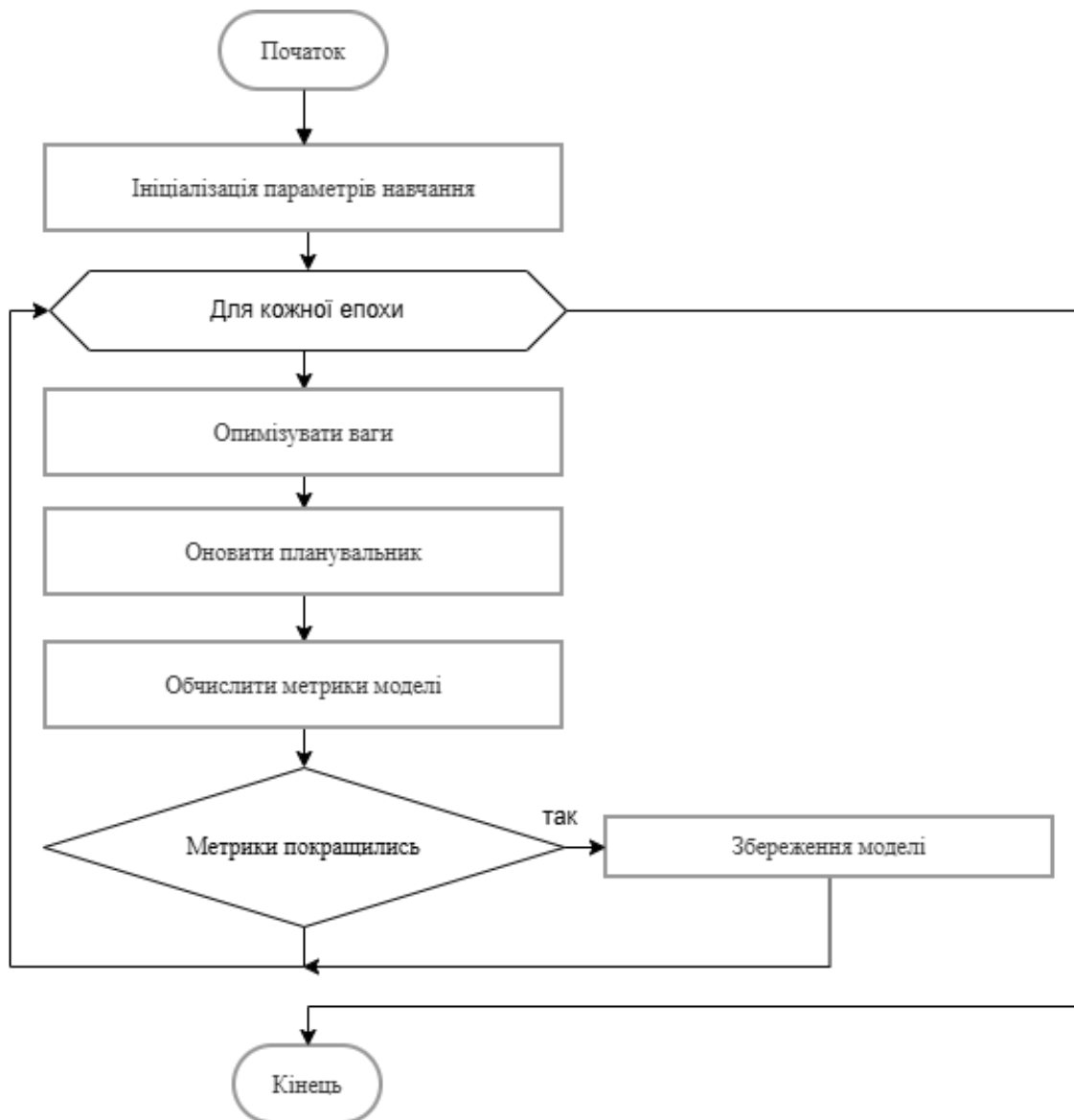


Рисунок 2.6 – Схема алгоритму другого етапу навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ

Процес починається з проходження тренувальних даних через модель у режимі оцінювання, щоб отримати вихідні значення для кожного зразка.

Функція втрат обчислюється для оцінювання ступеня помилки: це відсоток, на скільки вихідне значення для правильного класу нижче за максимальне (тобто, "скільки відсотків не вистачає до правильної класифікації"). Якщо модель передбачає неправильний клас (тобто, максимальний логіт відповідає нецільовому нейрону вихідного шару), то ініціюється корекція ваг. Для збуджувальних ваг (позитивних з'єднань від резервуарних нейронів до правильного вихідного нейрона) застосовується збільшення, пропорційне значенню функції втрат – це посилює внесок активних резервуарних нейронів, які повинні сприяти правильній класифікації. Для гальмівних ваг (негативних з'єднань до неправильних вихідних нейронів) значення втрат ділиться на кількість вихідних нейронів, і результат додається як більш негативне значення, що посилює гальмування небажаних активацій.

Обґрунтування цього етапу як фінального кроку для специфічного налаштування ваг полягає в його здатності адресувати залишкові помилки, які не були повністю виправлені глобальними методами. Градієнтний спуск ефективний для загальної оптимізації, але може ігнорувати рідкісні або незбалансовані випадки, особливо в задачах з високою варіабельністю даних, як аналіз ЕКГ-сигналів для детекції аритмій. Цей підхід забезпечує персоналізовану адаптацію: посилення збуджувальних ваг покращує чутливість для цільових класів, зменшуючи пропуски аритмій, тоді як посилення гальмівних ваг знижує помилкові позитивні результати, підвищуючи влучність. Він також сприяє біологічній правдоподібності моделі, імітуючи локальну пластичність синапсів на основі помилок, і є обчислювально ефективним, оскільки не вимагає повного зворотнього поширення помилки через весь резервуар, а лише корекцію вихідного шару. У контексті дисертації цей етап підкреслює гібридний підхід до навчання СНМ, комбінуючи статистичні, градієнтні та правило-орієнтовані методи для досягнення високої ефективності в медичних застосуваннях, де точність класифікації критична для життя пацієнтів. Крім того, поділ втрат на кількість вихідних нейронів нормалізує корекцію, запобігаючи надмірному впливу на

окремі класи в багатокласових задачах, і забезпечує стабільність мережі, уникаючи перетренування.

Цей етап може виконуватися на валідаційному або тренувальному наборі з контролем за метриками (наприклад, моніторинг F1-score або kappa-коефіцієнта), і його ітерації обмежуються, щоб уникнути переоптимізації. У результаті ваги стають більш адаптованими до специфічних патернів даних, покращуючи узагальнення моделі на небачених зразках.

Покроковий алгоритм 3-го етапу:

1) Перевести модель у режим оцінювання для фіксації параметрів резервуару та отримання стабільних виходів.

2) Ініціалізувати функцію втрат.

3) Для кожного зразка в тренувальному наборі даних:

- Виконати прямий прохід моделі.
- Порівняти передбачений з істинним класом: якщо передбачений клас не збігається з істинним, перейти до корекції; інакше – пропустити зразок.
- Обчислити значення функції втрат: визначити відсоток помилки (наприклад, (максимальний логіт – логіт правильного класу) / максимальний логіт, нормалізований до $[0; 1]$, де 1 означає повну помилку).

4) Для корекції збуджувальних ваг (позитивних з'єднань до правильного вихідного нейрона):

- Ідентифікувати збуджувальні ваги (ті, що > 0) від активних резервуарних нейронів до істинного класу.
- Збільшити кожен таку вагу на значення, пропорційне втраті (наприклад, вага $+=$ втрата * масштабний коефіцієнт, де масштаб – гіперпараметр для контролю швидкості оновлення).

5) Для корекції гальмівних ваг (негативних з'єднань до неправильних вихідних нейронів):

- Ідентифікувати гальмівні ваги від активних резервуарних

нейронів до передбаченого (неправильного) класу.

- Обчислити корекцію: розділити значення втрат на кількість вихідних нейронів (N_OUTPUT), щоб нормалізувати вплив.
- Зменшити кожен таку вагу (зробити більш негативною) на це значення (наприклад, $вага -= (втрата / N_OUTPUT) * \text{масштабний коефіцієнт}$).

6) Після обробки всіх зразків: нормалізувати ваги вихідного шару.

7) Оцінити модель на тестовому наборі: обчислити метрики для перевірки покращення.

8) Якщо метрики покращилися, зберегти оновлену модель.

Схему алгоритму третього етапу навчання наведено на рис. 2.7.

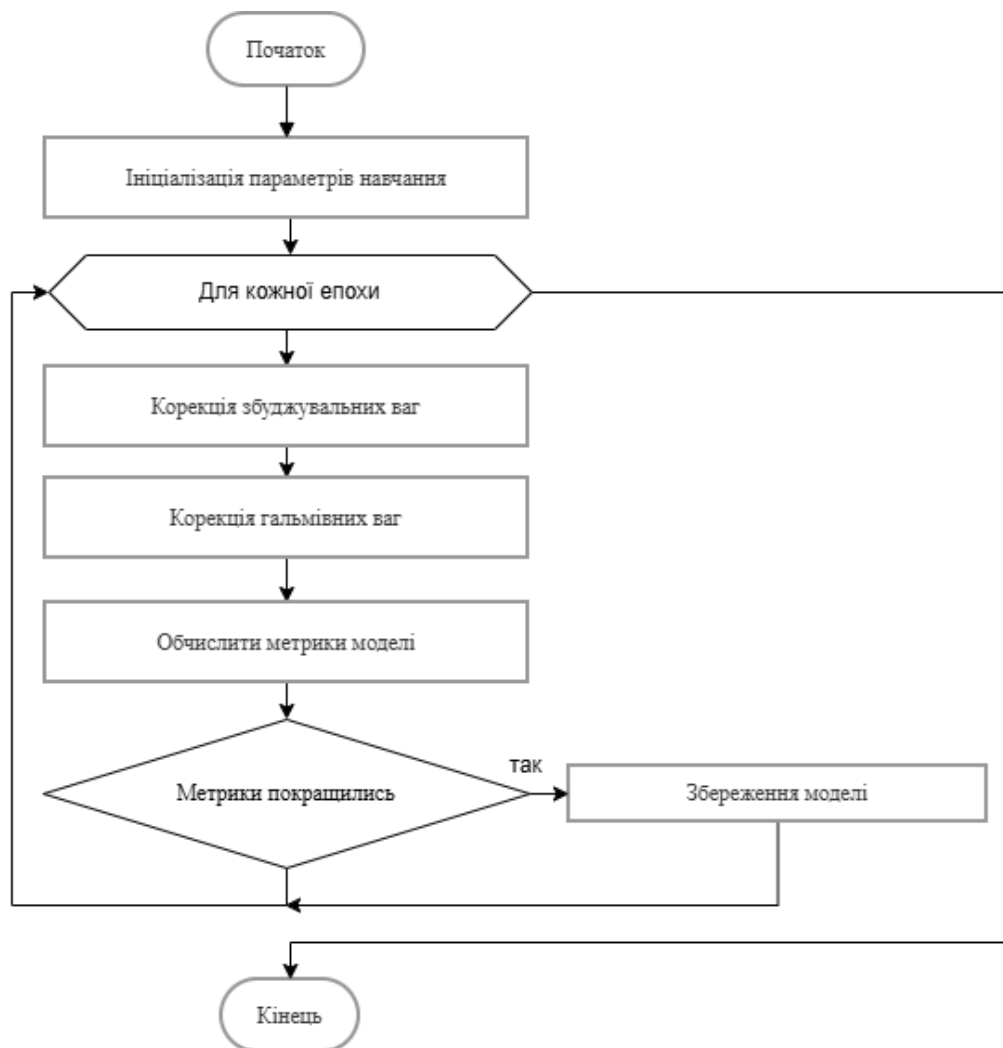


Рисунок 2.7 – Схема алгоритму третього етапу навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ

Завдяки такому багатоетапному тренуванню нейронна мережа буде мати досить високі метрики класифікації, що є метою навчання і максимізації чутливості – метрики, що має вирішальну вагу у оцінюванні методів класифікації у медичній сфері.

2.4 Розробка методу попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу

У рамках дисертаційного дослідження для забезпечення стандартизації вхідних даних, усунення шумових впливів, зумовлених індивідуальними фізіологічними особливостями пацієнтів, нестабільністю апаратури реєстрації та іншими факторами, розроблено комплексний метод попередньої обробки сигналу ЕКГ. Цей метод передбачає сегментацію сигналів ЕКГ, уніфікацію частоти дискретизації, амплітудну нормалізацію та часову нормалізацію. Підхід дозволяє отримати фіксовані за довжиною та масштабом сегменти, придатні для подальшого використання в спайкінговій нейронній мережі.

У межах розробленого методу здійснено передискретизацію до частоти дискретизації 500Гц, мінімаксну нормалізацію амплітуди та часову нормалізацію сигналу до 200 відліків та строге центрування R-піку (сигнал ділиться на 2 частини: 100 відліків до і 100 відліків після R-піку) даних електрокардіограми з бази даних MIT-BIH, де варіабельність форми сигналів ЕКГ зумовлена як індивідуальними фізіологічними особливостями суб'єктів спостереження, так і артефактами апаратного реєстрування, що ускладнює подальшу уніфікацію для навчання спайкінгових нейронних мереж. Початковим етапом первинної обробки даних постає систематичний прохід по файлу-анотації, який забезпечує точну ідентифікацію R-піків QRS-комплексів, а при обробці сирих сигналів можуть бути використані алгоритми детекції R-піків (наприклад, Pan-Tompkins), тим самим мінімізуючи обчислювальні витрати та підвищуючи надійність класифікації сигналів ЕКГ.

Метод складається з наступних етапів:

1. Сегментація сигналу ЕКГ навколо R-піків
2. Передискретизація (ресемплінг) сигналу до частоти 500 Гц
3. Мінімаксна нормалізація амплітуди в діапазон [0; 1]
4. Часова нормалізація сегментів
5. Кодування нормалізованого сигналу в Гаусові рецептивні поля

Узагальнена схема методу наведена на рис. 2.8.

На першому етапі здійснюється сегментація неперервного сигналу ЕКГ на окремі фрагменти (сегменти). Для кожного R-піку (який вважається референтною точкою сигналу) сегмент формується від середини інтервалу RR між попереднім і поточним R-піками до середини інтервалу RR між поточним і наступним R-піками. Такий підхід забезпечує неповторюваність сегментів, охоплює повний цикл серцебиття (включаючи Р-хвилю, комплекс QRS та Т-хвилю) та мінімізує перекриття між сусідніми сегментами.

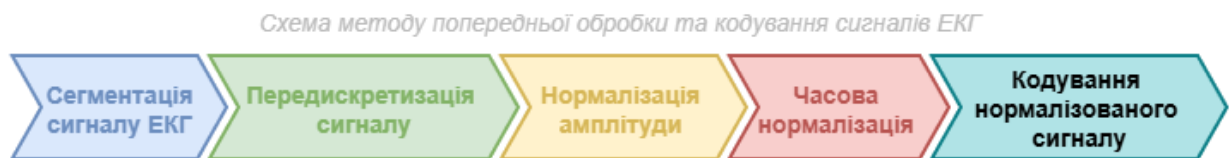


Рисунок 2.8 – Узагальнена схема методу попередньої обробки та кодування сигналу ЕКГ

Математично, якщо R_{i-1} , R_i , R_{i+1} , індекси відповідних R-піків, то сегмент для i -го кардіоциклу визначається як інтервал від $\frac{R_{i-1}+R_i}{2}$ до $\frac{R_i+R_{i+1}}{2}$. Передбачається, що R-піки вже виявлені за допомогою надійних алгоритмів.

Після сегментації сигнал передискретизується до єдиної частоти дискретизації 500 Гц за допомогою лінійної інтерполяції [83]. Цей крок уніфікує дані, отримані з різних джерел або пристроїв, і забезпечує сумісність із сучасними базами даних (наприклад, MIT-BIH, PTB-XL).

Наступний етап передбачає визначення мінімального та максимального значень сигналу в межах кожного інтервалу між R-піками. Цей крок є

підготовчим для подальшої мінімаксної нормалізації амплітуд. Сигнал розглядається як одновимірний масив значень напруги в мілівольтах. Обчислення $\min_s = \min(s)$ та $\max_s = \max(s)$, де s – вектор сигналу, дозволяють встановити межі для подальшої мінімаксної нормалізації, яка, застосована на рівні окремих сегментів, забезпечує перехід значень до інтервалу $[0; 1]$, усуваючи вплив абсолютних амплітудних варіацій, зумовлених відмінностями в апаратурі запису чи фізіологічними особливостями пацієнтів. На це можуть впливати товщина шкіри чи положення електродів, і цей крок, хоча й здається простим, є суттєвим для спайкінгових нейронних мереж, оскільки в них вхідні імпульси часто кодуються через частотне або часове кодування, де нормалізовані значення полегшують генерацію спайків з використанням моделі нейронів на кшталт LIF, зменшуючи шум і підвищуючи роздільну здатність моделі до дрібних морфологічних змін. Мінімальні та максимальні межі амплітуд наведено в таблиці 2.1, що показує варіабельність амплітудних відхилень поміж пацієнтами. Загальні межі амплітуд серед усіх записів MIT-BIH становить $[- 5.12; 5.115]$ мВ.

Таблиця 2.1 – Межі амплітуд по записам

№ запису	min, мВ	max, мВ	№ запису	min, мВ	max, мВ	№ запису	min, мВ	max, мВ
100	-2.715	1.435	113	-1.225	2.550	201	-1.315	2.045
101	-3.175	2.420	114	-2.390	1.815	202	-2.755	2.920
102	-1.660	1.375	115	-2.885	2.410	203	-4.305	3.640
103	-2.610	3.125	116	-5.120	5.115	205	-2.175	1.455
104	-1.935	1.975	117	-3.015	0.710	207	-3.955	2.285
105	-3.715	3.000	118	-3.090	2.395	208	-3.485	3.650
106	-2.065	2.570	119	-2.985	2.500	209	-1.455	1.885

Продовження табл. 2.1

№ запису	min, мВ	max, мВ	№ запису	min, мВ	max, мВ	№ запису	min, мВ	max, мВ
107	-3.315	3.465	121	-4.450	1.400	210	-1.620	3.320
108	-3.035	2.365	122	-2.200	1.250	212	-1.855	2.545
109	-3.195	3.130	123	-3.095	2.060	213	-2.530	3.285
111	-2.820	2.375	124	-3.605	2.595	214	-2.730	2.800
112	-2.635	0.820	200	-2.695	2.380	215	-1.380	2.500

Для усунення варіативності амплітуди, спричиненої різними умовами реєстрації, застосовується мінімаксна нормалізація кожного сегмента:

$$\tilde{x}[n] = \frac{x[n] - \min(x)}{\max(x) - \min(x)}, n = 0, 1, \dots, N - 1 \quad (2.5)$$

де $x[n]$ – значення амплітуди оригінального сегмента, $\tilde{x}[n]$ – нормалізоване значення в інтервалі $[0; 1]$. Такий підхід робить сигнал інваріантним до абсолютного масштабу та сприяє стабільнішому навчанню нейронних мереж.

Далі, для кожного сигналу застосовується часова нормалізація оскільки довжини сегментів можуть варіюватися від приблизно 200 до 600 відліків залежно від частоти серцебиття (наприклад, при нормальному синусовому ритмі близько 60-100 ударів на хвилину інтервал становить 0.6-1 секунду та зменшується при збільшенні ЧСС). Ця нормалізація виконується шляхом розширення або звуження до дистанції між серединам між сусідніми RR в 200 відліків та центруванням R-піку. Для цього:

- Частина сигналу до R-піку (пре-R) лінійно інтерполюється до 100 відліків.
- Частина сигналу після R-піку (пост-R) аналогічно

трансформується до 100 відліків.

- Отримані частини об'єднуються в єдиний вектор довжиною 200 відліків.

Лінійна інтерполяція реалізується за формулою [87]:

$$y(t) = y_0 + (t - t_0) * \frac{y_1 - y_0}{t_1 - t_0} \quad (2.6)$$

де t_0, y_0 – значення та час попереднього відомого відліку, t_1, y_1 – значення та час наступного відомого відліку, t – час для якого потрібно знайти значення відліку, $\frac{y_1 - y_0}{t_1 - t_0}$ – нахил прямої між точками.

Враховуючи комплексність взаємодій між часовими та амплітудними аспектами сигналу, запропонований метод попередньої обробки сигналу ЕКГ суттєво спрощує подальшу задачу класифікації оскільки нормалізує всі важливі данні та не втрачає корисну інформацію різних комплексів серцебиття.

Порівняння методів нейронного кодування сигналів ЕКГ доцільно проводити, виходячи з ефективності передачі морфологічної інформації у вхідний шар спайкінгової мережі та здатності зберегти її для подальшої обробки у резервуарі. Класичні підходи частотного та часового кодування мають принципово різні механізми роботи і демонструють різну чутливість до структури часових рядів [88], [89].

Частотне кодування інтерпретує інтенсивність вхідного стимулу як частоту спайкінгової активності: кожен нейрон генерує імпульси з частотою, пропорційною амплітуді сигналу. Формально можна розглядати його реалізацію через Пуассонів потік зі швидкістю λ , де λ пропорційна нормалізованому значенню $x(t)$. Такий метод добре відображає середньо- чи довготривалі зміни рівня стимуляції. Однак він потребує накопичення статистики спайків на часовому відрізку довжиною більше сталої часу мембрани τ_m , що обмежує роздільну здатність при швидких фазових

переходах (наприклад, круті фронти QRS). Через усереднення активності у часі частотне кодування має низький коефіцієнт передачі високочастотних компонент [88].

Часове кодування (Time-to-First-Spike, TTFS) кодує величину стимулу через затримку між моментом його подачі та першим спайком нейрона. Проста залежність може бути виражена як

$$t_{\text{spike}} = t_0 + f(x) \quad (2.7)$$

де $f(x)$ монотонно спадає при збільшенні амплітуди. У найпростішому випадку плаский фронт можна передати точно: сильний сигнал викликає ранній спайк, слабший - пізній або його відсутність [90]. Це дає високу точність для коротких імпульсів без використання багатьох спайків. Водночас TTFS погано масштабується на сигнали зі складною багатохвильовою структурою: повторні зміни амплітуди не можуть бути ефективно відображені після першого імпульсу. Таким чином значна частина часової морфології ЕКГ губиться, особливо якщо є кілька діагностично важливих піків (P-QRS-T).

Гаусові рецептивні поля (Gaussian Receptive Fields, GRF), представляють третій підхід, який інтегрує переваги локальної чутливості й популяційної надмірності. Тут кожен вхідний LIF-нейрон відповідає Гаусовим рецептивним профілем [91]:

$$g_i(x) = \exp\left(-\frac{(x - \mu_i)^2}{2\sigma^2}\right) \quad (2.8)$$

де μ_i – центри що відповідають значенням амплітуди сигналу, параметр σ задається так, щоб забезпечити одночасно хорошу локалізацію і достатнє перекриття для плавної передачі змін, що важливо для підтримки хвилі змін резервуару LSM, оскільки кожен момент активації стимулює нейрони, що відповідають сусіднім значенням центрів μ_i .

На відміну від частотного кодування, Гаусові рецептивні поля не вимагають великих вікон усереднення: навіть разове підвищення амплітуди активує відповідне поле і його сусідні профілі без втрат високочастотних деталей. На відміну від часового кодування, цей метод не обмежується одним імпульсом на хвилю: будь-які зміни вхідної амплітуди перетворюються на реакцію групи полів у реальному часі.

Цей підхід дозволяє змінювати τ_m для балансування між швидкою реакцією на QRS і довшою інтеграцією хвиль P/T. При частотному кодуванні будь-яке прискорене згасання через мале τ_m призведе до втрати інтенсивності кодування; при часовому — значення τ_m впливає лише на форму постспайкінгової відповіді, а сама інформація про пізні компоненти може бути вже недоступна після першого імпульсу. А Гаусові рецептивні поля завжди подають на нейрон актуальне значення активації незалежно від історії, отже доповнює часову функцію LIF без втрати деталей через обмеження алгоритму кодування.

З точки зору формування стану резервуару для вихідного шару, різниця у кодувальниках визначає якість відокремлення станів. Частотне кодування генерує переважно одномірний опис рівня середньої активації за період, тобто схожі за амплітудою сегменти можуть мати майже однаковий вихід незалежно від форми хвилі. Часове кодування створює сильну різницю лише за першими піками; але усі наступні патерни при близьких затримках у різних класах можуть накладатися й знижувати роздільність. Гаусові рецептивні поля генерують багатовимірну карту співактивацій полів: навіть мінімальні зміни локальної морфології змінюють конфігурацію активованих груп, ця варіативність поширюється рекурентними шляхами по резервуарному шару, створюючи виражене розділення траєкторій стану [77].

Важливо враховувати взаємодію кодувальника із попередньою обробкою даних. Часове та частотне кодування особливо чутливі до масштабування амплітуди й темпу, будь-яка помилка нормалізації змінює базовий параметр (частота чи час затримки) і може порушити відповідність

між класами та шаблонами виходу. Гаусові рецептивні поля менш залежать від глобальних масштабів завдяки фіксованому діапазону центрів μ_i й побудові їх на нормалізованій шкалі $[0; 1]$, тому стабільніше передає форму сигналу навіть при невеликих залишкових похибках інтерполяції чи мінімаксної нормалізації.

Таким чином у контексті класифікації ЕКГ Гаусові рецептивні поля утворюють більш стійкий та інформаційно насичений канал передачі морфології порівняно з частотним чи часовим кодуванням. Вони одночасно підтримують багатовекторний вплив на внутрішню динаміку резервуару та забезпечують адекватне представлення як швидких високочастотних компонент (QRS), так і повільних коливань (P/T) без матеріальних втрат діагностичної цінності [77], [89], [91].

2.5 Розробка методу класифікації сигналу ЕКГ на основі спайкінгової нейронної мережі

Враховуючи структуру спайкінгової нейронної мережі, а також розроблене комплексне правило навчання нейронів вихідного шару спайкінгової нейронної мережі, принципи роботи та математичні моделі LIF нейронів, можна сформулювати метод класифікації сигналів ЕКГ з використанням спайкінгової нейронної мережі.

Цей метод полягає в такому:

1. Створити рекурентну спайкінгову нейронну мережу, що функціонує за принципом резервуарного числення, складену з інтер-нейронів, в кількості не менше 1000. З'єднання нейронів в мережі виконати за законом Дейла та збереженням щільності в 10%. Ваги зв'язків нейронів вибрати невеликими випадковими та з відповідними знаками для збуджувальних та гальмівних зв'язків.

2. Сформувати масив вхідних нейронів, що будуть видавати результати кодування Гаусових рецептивних полів. З'єднати кожен з них випадковим

чином з не менше, ніж 20% нейронів резервуару. Ваги зв'язків вибрати випадковими збуджувальними.

3. Сформувати 12 вихідних нейронів (по 1 на кожен клас для ідентифікації типу серцебиття). З'єднати кожен з них випадково з не менше, ніж 20% нейронів резервуару. Ваги зв'язків вибрати випадковими зі збереженням закону Дейла.

4. Застосувати алгоритм навчання, описаний в п.2.3. Фактичний вихідний сигнал $f(x(t))$ буде послідовністю імпульсів, кількість яких, віднесена до максимальної показує на скільки (імовірність) вхідний сигнал відповідає фінальному класу.

5. Подати на вхід досліджуваний сигнал ЕКГ.

6. Нормалізувати та кодувати досліджуваний сигнал ЕКГ методами, що були описані у п 2.4. та подати нормалізований за амплітудою та довжиною сигнал ЕКГ тривалістю T_0 на вхід нейронної мережі і зафіксувати, який з вихідних нейронів буде мати максимальне значення кількості імпульсів протягом T_0 . Саме цей нейрон і визначає клас серцебиття, якому максимально відповідають вхідні сигнали мережі.

Запропонований метод має такі переваги:

1) завдяки використанню розробленого методу попередньої обробки ЕКГ сигналу, який містить амплітудну та часову нормалізацію та кодування за допомогою Гаусових рецептивних полів, вдається зменшити вплив на результат класифікації особливостей пацієнтів та апаратури зняття ЕКГ, що сприяє підвищенню точності класифікації сигналів ЕКГ спайкінговою нейронною мережею;

2) завдяки подачі сигналу ЕКГ на спайкінгову нейронну мережу для класифікації у вигляді часового ряду сталої довжини та амплітуди без попереднього перетворення у вектор статичних ознак, з одного боку, спрощується процес обробки через відсутність складних обчислювальних процедур перетворення, а з іншого боку, не втрачається через відсутність

перетворення корисна інформація, а тому – підвищується точність класифікації сигналів ЕКГ;

3) завдяки використанню розробленого методу навчання, який має три етапи, покращується якість навчання спайкінгової нейронної мережі, що в результаті призводить до підвищення точності класифікації сигналів ЕКГ.

2.6 Висновки до другого розділу

У цьому розділі було досліджено та сформовано теоретичні основи процесу класифікації сигналів ЕКГ на основі спайкінгових нейронних мереж. Обґрунтовано вибір моделі LIF нейрону та розроблено архітектуру спайкінгової нейронної мережі, яка відрізняється від відомих структурою шарів та параметрами з'єднання нейронів в резервуарі та резервуару із вхідним та вихідним шарами.

Запропоновано метод навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ. Особливістю цього методу є додавання етапів кластерної ініціалізації ваг вихідних нейронів та керованої спайками корекції збуджувальних ваг цільового вихідного нейрона та гальмівних ваг інших нейронів, що забезпечило покращення якості навчання спайкінгової нейронної мережі та підвищення точності класифікації сигналів ЕКГ

Розроблено метод попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу. До запропонованих змін входить введення часової нормалізації сегментів та кодування нормалізованого сигналу в Гаусові рецептивні поля. Такий підхід забезпечує зменшення впливу на результат особливостей пацієнтів та апаратури зняття ЕКГ, що сприяє підвищенню точності класифікації сигналів ЕКГ спайкінговою нейронною мережею.

Розроблено метод класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі, який має ряд переваг перед відомими методами, зокрема, забезпечує інваріантність до фізіо-електричних особливостей пацієнта та

апаратури зняття ЕКГ, не вимагає попереднього перетворення у вектор статичних ознак, має спрощений процес обробки через відсутність складних обчислювальних процедур, має покращену якість навчання спайкінгової нейронної мережі, може класифікувати сигнали ЕКГ в більшу кількість класів, ніж відомі методи, що в результаті призводить до підвищення точності класифікації сигналів ЕКГ за збереження ресурсоефективності та реально-часової швидкодії.

Основні наукові результати, викладені у розділі, опубліковано в роботах [1], [3], [4], [7], [8].

РОЗДІЛ 3

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ НА ОСНОВІ СПАЙКІНГОВИХ НЕЙРОННИХ МЕРЕЖ

3.1 Загальна схема та етапи інформаційної технології

Запропонований метод класифікації сигналу ЕКГ на основі спайкінгової нейронної мережі може бути представлений у вигляді інформаційної технології, що має 2 режими: навчання та класифікація. Кожен з режимів має свою логіку роботи та алгоритм дій. Отже, загальна схема інформаційної технології буде мати вигляд, як на рис. 3.1.

У якості вхідних даних для режимів роботи інформаційної технології використовуються:

- Для режиму навчання – набір анотованих даних ЕКГ з бази МІТ-ВІН Arrhythmia Database, який використовується для навчання та тестування моделі.
- Для режиму використання – сирі (необроблені) дані ЕКГ пацієнта, отримані в реальному часі або з портативного монітора.

Далі йде етап попередньої обробки та кодування сигналу ЕКГ, який майже однаковий для 2 режимів роботи. Спочатку виконується обесшумлення та віднімання постійної складової. Наступний етап відрізняється визначенням R-піків: для режиму навчання використовуються вже анотовані положення з датасету, для режиму використання – можуть бути застосовані відомі методи визначення R-піків (наприклад, Пан-Томкінс). Далі виконується амплітудна та часова нормалізації. Амплітуда приводиться до інтервалу $[0; 1]$, а сегменти вирівнюються до довжини в 200 відліків та з центруванням R-піку. Останнім етапом є перетворення нормалізованого часового ряду ЕКГ у послідовність спайків за допомогою Гаусових рецептивних полів. Цей метод дозволяє ефективно представляти аналогову інформацію у формі часової та просторової розподіленості імпульсів.

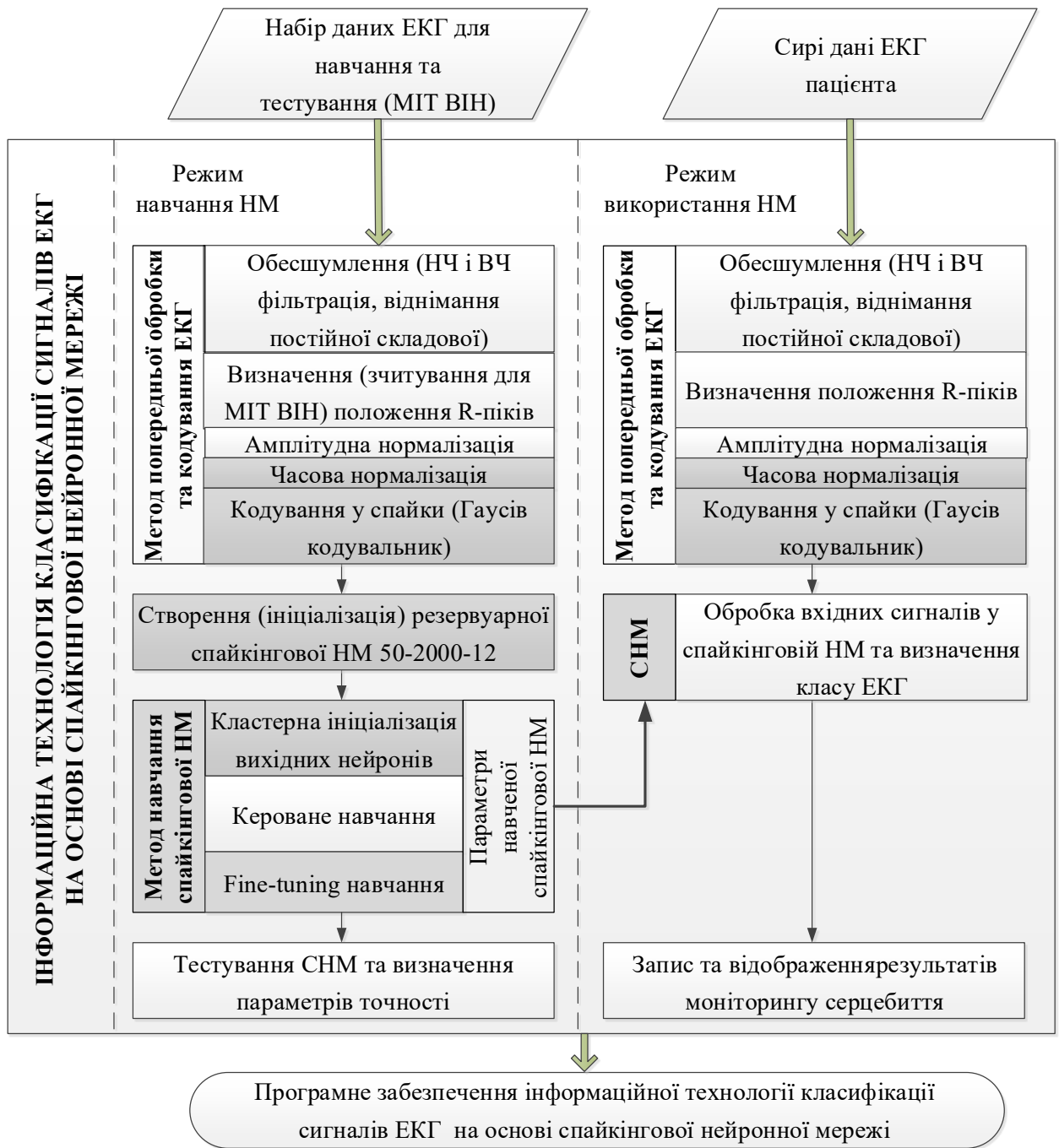


Рисунок 3.1 – Загальна схема інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі

Наступним етапом інформаційної технології є обробка вхідних кодованих даних сигналу ЕКГ і відрізняється для режимів роботи інформаційної технології.

Якщо обрано режим тренування, то інформаційна технологія створить нову нейронну мережу з новим набором параметрів – ваг зв'язків між нейронам та індексів нейронів, які пов'язані, і почне 3-етапне тренування. За

бажанням в кінці тренування можна отримати метрики, що потрібні для порівняння продуктивності з іншими системами. В результаті отримані параметри навченої спайкінгової нейронної мережі зберігаються для подальшого використання у режимі використання.

Якщо обрано режим використання, то інформаційна технологія задіє навчену нейронну мережу для класифікації та отримання результату, за бажанням можна отримати графік станів нейронної мережі.

3.2 Обґрунтування вибору набору даних для навчання та тестування спайкінгової нейронної мережі

MIT-BIH Arrhythmia Database є одним із найвідоміших та найцитованіших ресурсів у дослідженнях автоматичного аналізу ЕКГ, який використовують для тестування алгоритмів класифікації серцевих ударів [92]. Набір містить 48 двоканальних амбулаторних записів тривалістю приблизно півгодини кожен, з частотою дискретизації 360 Hz, що дозволяє аналізувати як часові, так і спектральні характеристики сигналу на високому рівні деталізації. Записи виконані за допомогою модифікованого відведення Limb Lead II (MLII) шляхом розміщення електродів на грудній клітці, що дає стабільну морфологію комплексу QRS і полегшує розпізнавання ЕКГ для різних пацієнтів [93]. Ці параметри апаратного налаштування особливо зручні для задач багатокласової класифікації, адже MLII вважається менш чутливим до певних артефактів, ніж стандартні кінцівкові відведення. Анотації бази виконані кардіологами-експертами та містять інформацію про понад 110000 серцевих ударів з їхнім віднесенням до різних категорій. Це забезпечує можливість аналізу як нормального синусового ритму, так і патологічних станів, від надшлункових скорочень до блокад ніжок пучка Гіса та інших складних форм аритмій. Детальні характеристики датасету наведені в табл. 3.1, а демографічне розподілення у табл. 3.2.

Таблиця 3.1 – Загальні технічні характеристики MIT-BIH Arrhythmia Database

Кількість записів	48 (по 30 хвилин кожен)
Кількість пацієнтів	47 (записи 201 та 202 – від одного чоловіка)
Загальна тривалість даних	≈ 24 години
Кількість каналів	2
Основні відведення	MLII, V1 (інколи V2, V4 або V5)
Частота дискретизації	360 Гц
Роздільна здатність	11-бітна, діапазон ±10 мВ
Рік запису	1975-1979
Загальна кількість анотацій	≈ 110000
Кількість анотованих серцевих скорочень	109494

Таблиця 3.2 – Демографічні характеристики пацієнтів

Чоловіки	25 осіб (вік 32-89 років)
Жінки	22 особи (вік 23-89 років)
Загальна кількість пацієнтів	47
Статус пацієнтів	≈60% стаціонарні, ≈40% амбулаторні (змішана популяція)
Особливості	Включає як нормальні ритми, так і рідкісні клінічно значущі аритмії

Категоризація в рамках стандарту AAMI включає класи N (норма), SVEB (суправентрикулярний ектопічний удар), VEB (вентрикулярний ектопічний удар), F (злиття ударів) та Q (невідомий тип) [94]. Наявність таких чітко визначених класів дає змогу використовувати базу як основу для навчання моделей навіть при значному розмаїтті морфології сигналу. Інша ключова характеристика – присутність шумів та артефактів у записах. Це не випадковість, а важливий аспект набору, оскільки він відображає реальні умови збору даних. MIT-BIH включає сигнали з базовим дрейфом, інтерференцією від електродів, впливом руху пацієнта тощо [95]. Для моделювання стійкості алгоритмів класифікації цей фактор є корисним: він

дозволяє перевіряти здатність системи працювати в умовах непередбачуваних спотворень.

Структура анотацій охоплює не лише мітки серцевих ударів і їх типи, але й позначення не-бітових подій (наприклад, шумових сегментів або маркерів технічних артефактів) [94]. Таке додаткове маркування може бути використане при побудові систем попередньої обробки для виключення ділянок сигналу, які не несуть клінічно значущої інформації. Кожен запис супроводжується синхронною інформацією з другого каналу, хоча в багатьох дослідженнях використовується лише перший канал MLII через його стандартизованість [92]. Важливо враховувати проблему дисбалансу класів у наборі: нормальні удари складають більшість випадків. Для мінімізації цієї проблеми часто застосовують методи балансування вибірки. Для архітектур глибокого навчання база добре пасує не лише через розмітку R-піків і високу якість запису, але й через свою варіативність форм сигналів та структурованість.

3.3 Дослідження варіабельності форми сигналів ЕКГ різних класів аритмії

Варіабельність морфології сегментів ЕКГ є фундаментальною характеристикою для розуміння механізмів аритмій та оцінюванні якості автоматичної класифікації. У базі даних MIT-BIH Arrhythmia Database сигнали вирівнюються (зазвичай по піку R-хвилі), нормалізуються за амплітудою (0 ; 1) та відображаються як накладені індивідуальні удари (тонкі сині лінії) навколо середнього удару (чорна пунктирна лінія). На рис. 3.4-3.15 кожен графік охоплює ~200 відліків часу, що дозволяє візуалізувати розкид у P-, QRS-, ST- та T-сегментах. Низька варіабельність (щільний пучок) свідчить про стабільну провідність і повторюваність, висока – про гетерогенність походження імпульсу, варіабельність інтервалу зчеплення, аберацію провідності, компенсаторні паузи або артефакти вирівнювання.

Аналіз проводиться для 12 класів (відповідність міткам у графіках і стандартній анотації MIT-BIH):

Class 0: Normal (N) – нормальне серцебиття (рис. 3.2). Середній удар – класична морфологія синусового ритму: вузький гострий QRS (~80-100 мс), чітка Р-хвиля перед QRS, нормальна позитивна Т-хвиля. Варіабельність мінімальна – лінії утворюють надзвичайно щільний пучок з практично нульовим розкидом у всіх фазах. Фізіологічно це пояснюється стабільним синусовим вузлом і нормальною провідною системою.

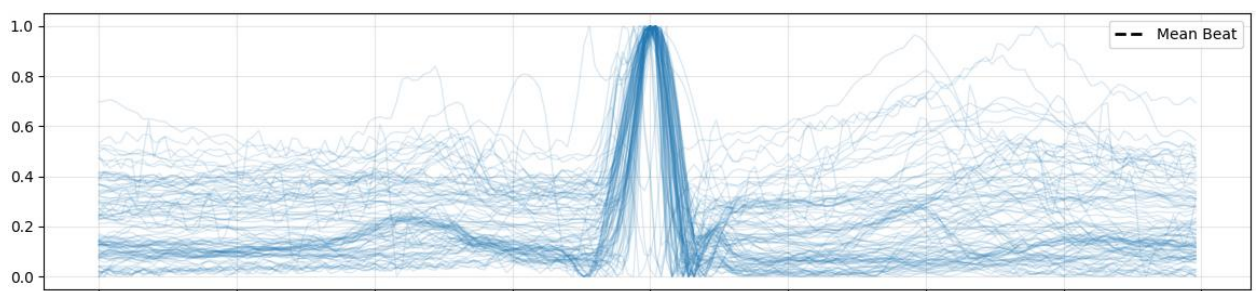


Рисунок 3.2 – Варіабельність форми сигналів ЕКГ для класу N

Class 1: Left bundle branch block (L) – блокада лівої ніжки пучка Гіса (рис. 3.3). Середній удар демонструє характерний широкий QRS (>120 мс) з однофазним або зазубреним R у бічних відведеннях. Варіабельність низька-середня: пучок дещо ширший за нормальний сигнал, особливо у висхідній/спадній частині QRS та ST-сегменті, але все ще щільний. Стабільність морфології зумовлена фіксованою блокадою лівої ніжки; невеликий розкид виникає через варіації частоти серцевих скорочень та незначні зміни вектора деполяризації.

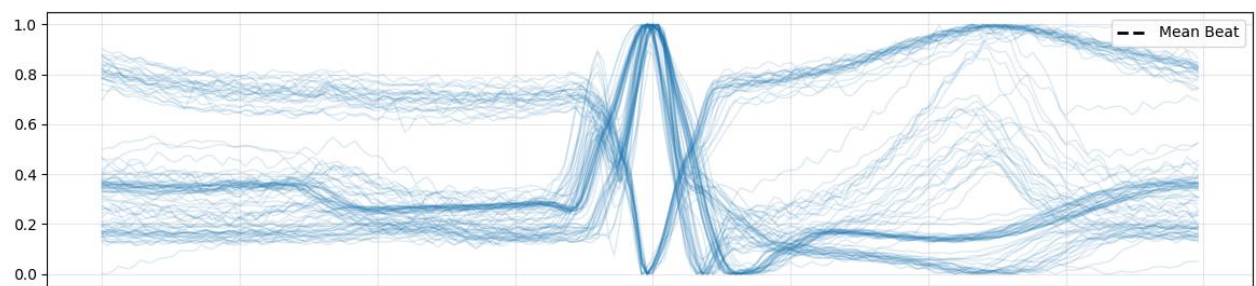


Рисунок 3.3 – Варіабельність форми сигналів ЕКГ для класу L

Class 2: Right bundle branch block (R) – блокада правої ніжки пучка Гіса (рис. 3.4). Широкий QRS з типовою морфологією у правих відведеннях. Варіабельність низька, подібна до блокади лівої ніжки пучка Гіса, але з трохи більшим розкидом у пізній R'-хвилі та Т-хвилі (часто інверсійна). Стабільність пояснюється фіксованою затримкою правої ніжки.

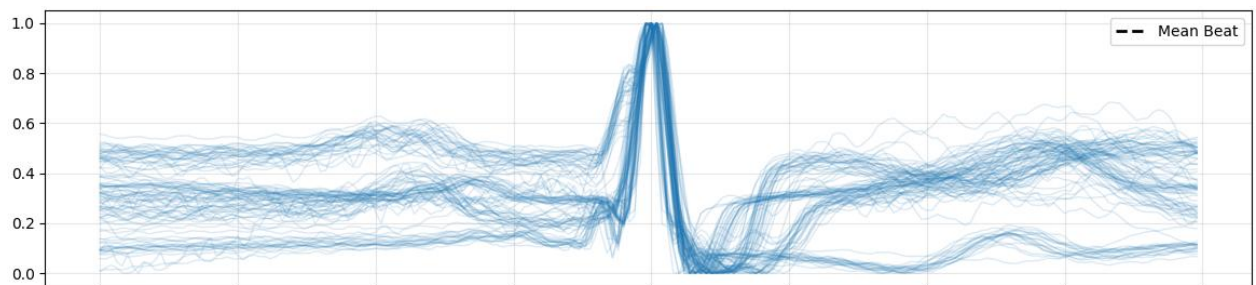


Рисунок 3.4 – Варіабельність форми сигналів ЕКГ для класу R

Class 3: Atrial escape (e) – передсердний замісний ритм (рис. 3.5). Середній удар близький до нормального за шириною QRS, але Р-хвиля відсутня або ретроградна/зміщена, передчасність мінімальна. Варіабельність середня: помітний розкид у пре-QRS зоні (відсутність/варіабельність Р) та невеликий у Т. Розкид виникає через варіабельність моменту виникнення імпульсу в передсердях та компенсаторну паузу попереднього циклу.

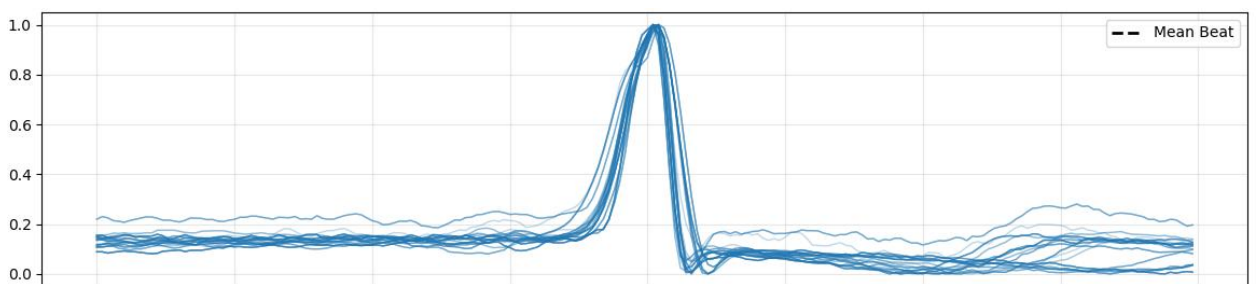


Рисунок 3.5 – Варіабельність форми сигналів ЕКГ для класу e

Class 4: Nodal escape (j) – вузловий замісний ритм (рис. 3.6). QRS нормальної ширини, Р-хвиля відсутня або ретроградна (інверсійна у нижніх відведеннях). Варіабельність середня-низька: щільніший пучок ніж у

передсердного замісного ритму, але з розкидом у пре-QRS (різні ступені ретроградної провідності) та пост-QRS (Т-хвиля).

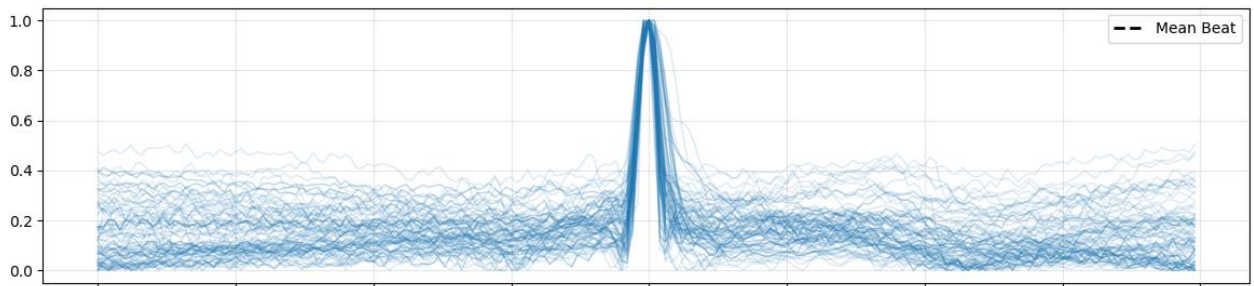


Рисунок 3.6 – Варіабельність форми сигналів ЕКГ для класу j

Class 5: Atrial premature (A) – передсердні екстрасистоли (рис. 3.7).

Передчасний удар з зміненою Р-хвилею (морфологія/амплітуда), нормальним QRS і коротким PR. Варіабельність середня: значний розкид у пре-QRS зоні (різні морфології Р та PR-інтервали) при відносно стабільному QRS і Т. Компенсаторна пауза після удару сприяє варіабельності базової лінії на наступних сегментах.

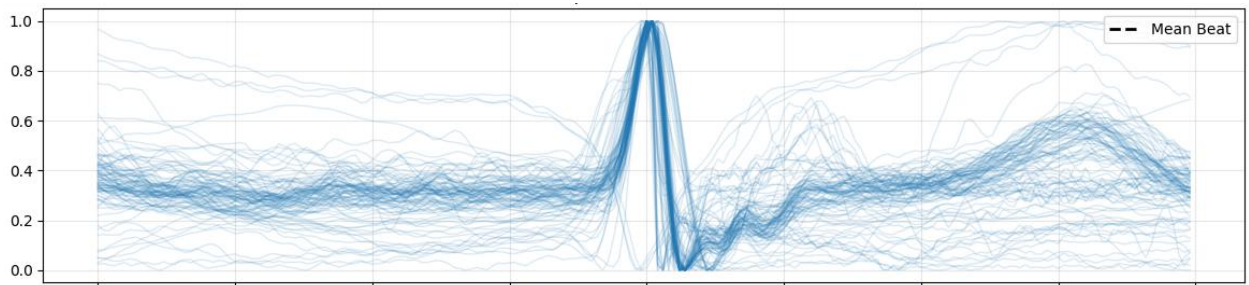


Рисунок 3.7 – Варіабельність форми сигналів ЕКГ для класу А

Class 6: Aberrated atrial premature (a) – передсердна екстрасистола з аберантним проведнням (рис. 3.8). Передчасний зубець Р удар з аберантною шлуночковою провідністю – широкий QRS, що нагадує блокаду пучка Гіса. Варіабельність висока: широкий розкид у QRS-комплексі (варіабельність ступеня аберації залежно від передчасності та рефрактерності), а також у ST-інтервалі. Це один із найваріабельніших

суправентрикулярних класів через динамічну залежність аберації від довжини інтервалу зчеплення.

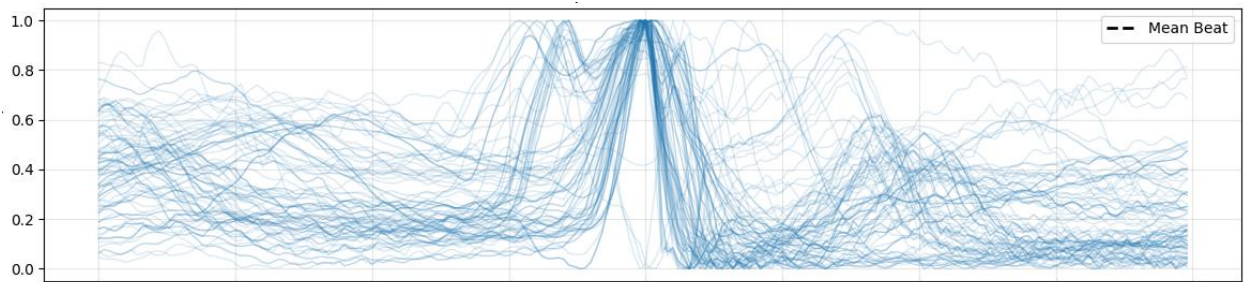


Рисунок 3.8 – Варіабельність форми сигналів ЕКГ для класу а

Class 7: Nodal premature (J) – вузлова передчасна екстрасистола (рис. 3.9). Передчасний удар з вузловим походженням – відсутність/ретроградна P, нормальний QRS. Варіабельність середня: розкид переважно у пре-QRS (різні ретроградні P) та T-хвилі; QRS стабільний. Подібний до вузлового замісного ритму, але з більшою передчасністю.

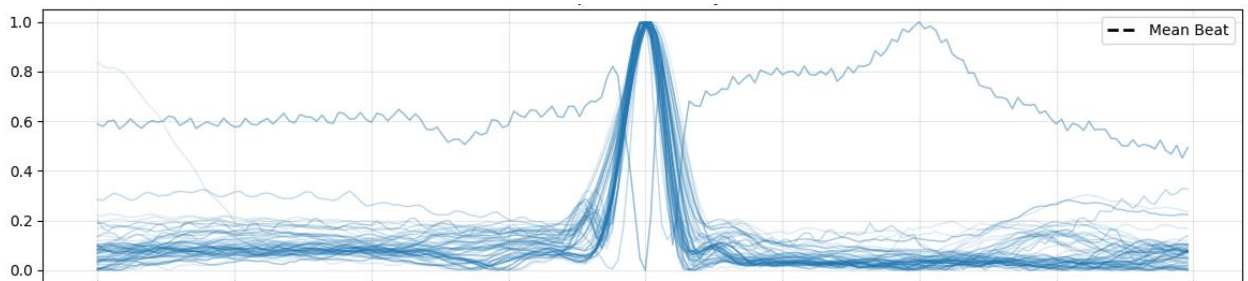


Рисунок 3.9 – Варіабельність форми сигналів ЕКГ для класу j

Class 8: Supraventricular premature (S) – надшлуночкова (суправентрикулярна) передчасна екстрасистола (рис. 3.10). Загальна категорія суправентрикулярних передчасних ударів (часто з нечіткою P або без P). Варіабельність середня-висока: помітний розкид у пре-QRS та PR, стабільний QRS.

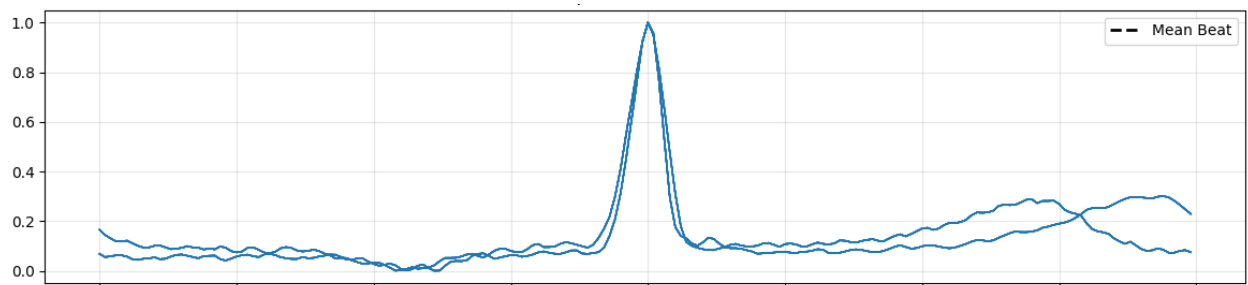


Рисунок 3.10 – Варіабельність форми сигналів ЕКГ для класу S

Class 9: Premature ventricular contraction (V) – шлуночкова екстрасистола (рис. 3.11). Широкий (>120 мс) аномальний QRS без передуючої Р (або з ретроградною), Т-хвиля часто протилежної полярності. Варіабельність висока: значний розкид у морфології QRS (різні точки виходу з міокарда шлуночків, різні шляхи проведення), ST-T сегменті та амплітуді. Один із найваріабельніших класів; варіабельність посилюється мультиморфністю сигналів ЕКГ у одного пацієнта.

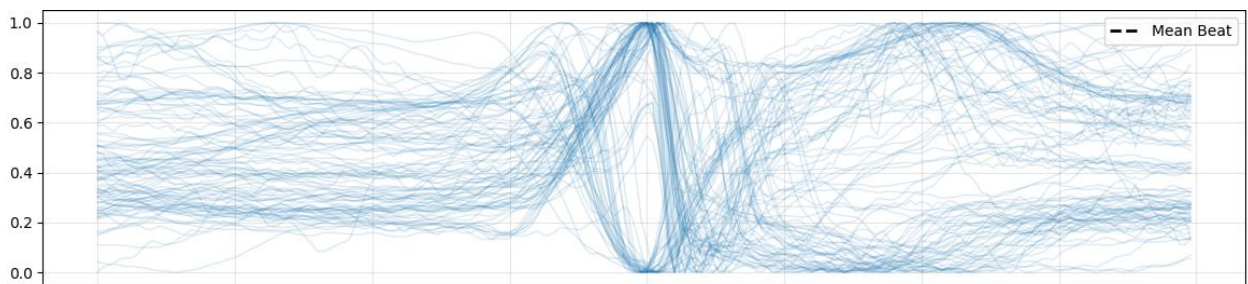


Рисунок 3.11 – Варіабельність форми сигналів ЕКГ для класу V

Class 10: Ventricular escape (E) – шлуночковий замісний ритм (рис. 3.12). Шлуночковий удар після довгої паузи – широкий аномальний QRS, подібний до шлуночкової екстрасистоли. Варіабельність висока: розкид у QRS-морфології та особливо у пост-QRS (Т, базова лінія), зумовлений варіабельністю паузи та автоматичності ектопічного фокусу.

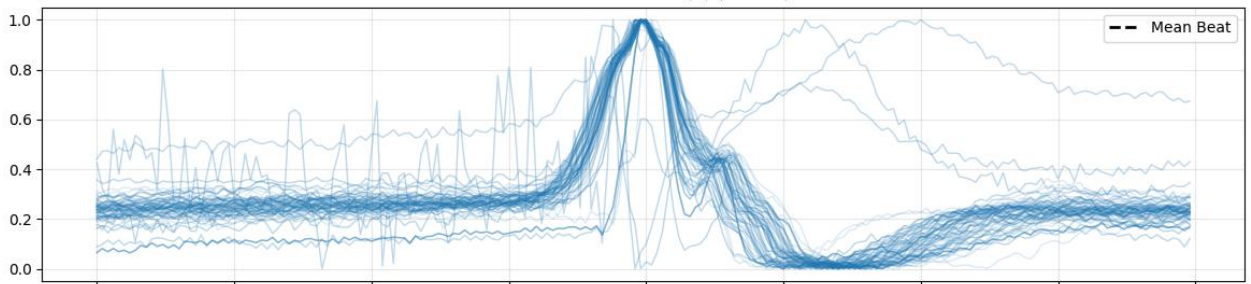


Рисунок 3.12 – Варіабельність форми сигналів ЕКГ для класу E

Class 11: Fusion of ventricular and normal (F) – змішаний або гібридний комплекс (рис. 3.13). Гібридна морфологія: часткове злиття нормального та шлуночкового імпульсів. Варіабельність дуже висока: лінії утворюють широкий спектр – від майже нормального QRS до майже вентрикулярного залежно від відносного таймінгу двох фронтів деполяризації.

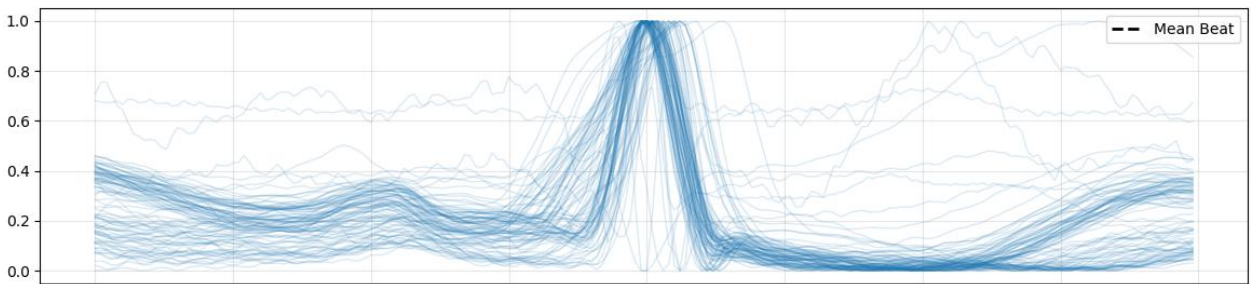


Рисунок 3.13 – Варіабельність форми сигналів ЕКГ для класу F

Загальні закономірності та висновки. З рисунків 3.2-3.13 видно, що найнижча варіабельність характерна для класів N, L, R – стабільні фіксовані шляхи провідності. Суправентрикулярні передчасні (A, J, S) демонструють розкид переважно у пре-QRS зоні. Аберантні (a) та всі шлуночкові класи (V, E, F) – найвищу варіабельність через варіабельність походження імпульсу, інтервалу зчеплення, рефрактерність та можливу мультиморфність. Замісні удари (e, j) займають проміжне положення.

Ця варіабельність безпосередньо впливає на продуктивність моделей глибокого навчання: класи з низькою варіабельністю (N, L, R) класифікуються з точністю >98 %, тоді як V, F, а потребують спеціальних технік аугментації,

передискретизації, доповнення форми сигналу (викривлення часу, масштабування амплітуди), застосування робастних функцій втрат або багатоканального аналізу. Висока внутрішньокласова варіабельність у вентрикулярних аритміях пояснює часті помилки між $V \leftrightarrow F \leftrightarrow L/R$ (морфологічна подібність).

Загалом, представлена візуалізація чітко демонструє, що варіабельність форми сигналів ЕКГ не є шумом, а відображає фізіологічну сутність аритмії. Її кількісна оцінка (наприклад, стандартне відхилення по відліках, DTW-відстань до еталонного удару) повинна обов'язково враховуватися при розробці та валідації класифікаторів.

Для подальшої обробки даних спайкінговою нейронною мережею було проведено ресемплінг даних до 2500 записів під кожен з 12 класів.

3.4 Вибір бібліотек та програмного середовища для реалізації інформаційної технології класифікації сигналів ЕКГ

Класифікація сигналів електрокардіограми (ЕКГ) є складним завданням штучного інтелекту та біомедичної інженерії, що поєднує обробку часових рядів, виявлення патологічних закономірностей та високу чутливість до помилок класифікації, особливо при виявленні рідкісних аритмій. Сучасні підходи до розв'язання цієї задачі дедалі частіше використовують спайкінгові нейронні мережі, які імітують біологічну динаміку нейронів і потенційно забезпечують вищу енергоефективність при реалізації на нейроморфному апаратному забезпеченні.

Основні вимоги до програмного середовища для реалізації такої технології можна сформулювати наступним чином:

- підтримка ефективної роботи з часовими послідовностями великої довжини (від кількох сотень до кількох тисяч відліків на один комплекс QRS);

- можливість реалізації динамічних нейронних мереж з рекурентними зв'язками та спайкінговою динамікою (LIF-нейрони, мембранний потенціал, порогове спрацьовування);
- наявність механізмів автоматичного диференціювання через час (BPTT) для навчання CHM;
- підтримка прискорення обчислень на GPU (CUDA) та, бажано, на Apple Silicon (MPS);
- зручні засоби завантаження та препроцесингу медичних баз даних (зокрема MIT-BIH Arrhythmia Database);
- розвинена екосистема для оцінювання якості класифікації на імбалансних даних (macro-recall, Cohen's kappa, PR-криві, ROC-аналіз);
- можливість візуалізації спайкінгової активності, стану резервуару та вихідних логітів для інтерпретації моделі.

Окремим важливим аспектом є вибір між традиційними штучними нейронними мережами (ANN) та спайкінговими (SNN). ANN зазвичай демонструють вищу точність при стандартному оцінюванні завдяки простоті навчання методом зворотного поширення помилки. Водночас SNN мають низку теоретичних та практичних переваг: біологічна правдоподібність, потенційно менша кількість операцій на інференс (керовані подіями обчислення), краща адаптація до імпульсних сенсорів та перспективність перенесення на нейроморфні чіпи (наприклад, Intel Loihi [42]). Саме тому в даній роботі обрано гібридний підхід на базі резервуарних мереж з вихідним шаром, реалізований у рамках спайкінгової парадигми.

Таким чином, програмне середовище повинно поєднувати гнучкість дослідницької платформи з достатньою продуктивністю для навчання та оцінювання моделей на повному наборі даних MIT-BIH (понад 100 тис. анотованих комплексів).

На сьогодні основними мовами програмування для реалізації алгоритмів машинного навчання та обробки біомедичних сигналів є Python, MATLAB, R, Julia та C++.

Python [96] є беззаперечним лідером у сфері глибокого навчання завдяки величезній екосистемі бібліотек (PyTorch, TensorFlow, NumPy, SciPy, wfdb, scikit-learn), великій кількості готових реалізацій моделей SNN та активній спільноті. Python дозволяє швидко розробляти задачі, легко інтегрувати різні інструменти та публікувати результати у наукових статтях. Недоліки – відносно нижча швидкість виконання порівняно з компільованими мовами, однак для більшості дослідницьких задач це компенсується використанням GPU-прискорення.

MATLAB [97] залишається популярним у біомедичній інженерії завдяки вбудованим інструментам обробки сигналів (Signal Processing Toolbox, Wavelet Toolbox) та зручному інтерфейсу для візуалізації. Проте висока вартість ліцензії, менша гнучкість у реалізації складних SNN та слабша підтримка сучасних фреймворків глибокого навчання роблять його менш привабливим для даної задачі.

Julia [98] позиціонується як високопродуктивна мова для наукових обчислень, що поєднує синтаксис Python з продуктивністю C. Проте екосистема для SNN у Julia (наприклад, Spike.jl) значно менш розвинена, ніж у Python.

C++ [99] використовується переважно для продуктивних інференс-додатків або реалізації на вбудованих системах, але розробка та налагодження моделей у ньому суттєво повільніші. Результати порівняння представлені у табл. 3.3.

Серед сучасних фреймворків глибокого навчання основними конкурентами є PyTorch та TensorFlow/Keras. PyTorch вирізняється динамічним графом обчислень, що значно полегшує реалізацію рекурентних структур, кастомних шарів та surrogate gradient методів, необхідних для навчання SNN. TensorFlow, попри потужну екосистему розгортання

(TensorFlow Lite, TensorFlow.js), менш зручний для дослідницьких експериментів з динамічними мережами.

Таблиця 3.3 – Порівняння програмних середовищ

Критерій	Pytho	MATLAB	Julia	C++
Швидкість розробки	Висока	Висока	Середня	Низька
Екосистема SNN	Відмінна	Слабка	Середня	Слабка
Підтримка GPU	Відмінна	Середня	Хороша	Хороша
Вартість	Безкоштовно	Висока	Безкоштовно	Безкоштовно
Інтеграція з wfdb/MIT-BIH	Відмінна	Хороша	Слабка	Слабка

Для реалізації саме спайкінгових нейронних мереж сьогодні доступні кілька спеціалізованих бібліотек:

- SpikingJelly (на базі PyTorch) – найбільш розвинена, підтримує LIF, IF, PLIF нейрони, surrogate градієнти (ATan, Sigmoid), рекурентні контейнери, інтеграцію з torchvision;
- Norse (PyTorch) – орієнтована на біологічну правдоподібність, але менш зручна для великих мереж;
- Brian2 – класична біологічно-реалістична симуляція, однак повільна для великих мереж і не підтримує автоматичне диференціювання;
- Lava (Intel) – орієнтована на Loihi, але наразі обмежена в функціональності для досліджень.

У даній роботі обрано SpikingJelly завдяки:

- повній сумісності з PyTorch: можливість використовувати всі інструменти PyTorch (AdamW, CosineAnnealingLR, DataLoader тощо);
- наявності готових класів для LIF-нейронів та рекурентних шарів (LinearRecurrentContainer);

- підтримці `detach_reset` та `surrogate` градієнтів для стабільного навчання;
- великій кількості прикладів реалізації резервуарних мереж та LSM-подібних архітектур.

Таким чином, комбінація Python, PyTorch, SpikingJelly забезпечує оптимальний баланс між гнучкістю, продуктивністю та можливістю реалізації ефективної спайкінгової моделі для класифікації ЕКГ.

Одним із найважливіших етапів створення системи автоматичної класифікації ЕКГ є якісна підготовка даних. База MIT-BIH Arrhythmia Database залишається золотим стандартом для тестування алгоритмів детекції аритмій завдяки великій кількості записів, чітким анотаціям та різноманітності патологічних станів. Однак сирі дані цієї бази потребують значної попередньої обробки: сегментації за комплексами QRS, нормалізації, вирівнювання довжини сегментів, балансування класів та адаптації під вимоги спайкінгової моделі.

Для зчитування даних обрано бібліотеку `wfdb` [100] (WaveForm DataBase) – офіційний інструмент PhysioNet для роботи з фізіологічними записами. Вона забезпечує:

- автоматичне завантаження бази даних з сервера PhysioNet за допомогою функції `wfdb.dl_database('mitdb', db_path)`;
- зчитування сигналів у вигляді масивів NumPy (`wfdb.rdsamp`) з метаданими (частота дискретизації 360 Гц, кількість каналів);
- зчитування анотацій (`wfdb.rdann`) з інформацією про тип кожного удару (символи N, V, S, F тощо) та точні позиції R-піків (`ann.sample`).

Нормалізація амплітуди проводиться за допомогою власної функції `min-max` у межах кожного сегмента, що усуває вплив різної амплітуди у різних пацієнтів.

Балансування класів реалізовано за допомогою `sklearn.utils.resample` з параметром `replace=True` для рідкісних класів (S, V, F), що дозволяє довести кількість зразків до 2500 на кожен клас.

Альтернативи, такі як `Biosppy` [101] та `HeartPy` [102], пропонують автоматичну детекцію QRS (алгоритм Пан-Томпкінс), фільтрацію базової лінії та видалення ознак. Однак у випадку MIT-BIH використання готових детекторів недоцільне, оскільки анотації вже містять точні позиції R-піків. Крім того, автоматичні алгоритми часто помиляються на зашумлених записах (наприклад, 108, 119), що призводить до втрати якості даних. Результати порівняння представлені у табл. 3.4.

Таблиця 3.4 – Порівняння бібліотек обробки ЕКГ

Бібліотека	Читання MIT-BIH	Точна сегментація за анотаціями	Інтерполяція	Балансування	Гнучкість під SNN	Вибрано
wfdb	Відмінна	Так	Ні	Ні	Висока	Так
Biosppy	Хороша	Ні (автодетекція)	Так	Ні	Середня	Ні
HeartPy	Середня	Ні (автодетекція)	Так	Ні	Низька	Ні
NeuroKit2 [103]	Хороша	Ні (автодетекція)	Так	Ні	Середня	Ні

Оцінювання якості класифікації аритмій вимагає комплексного підходу: числові метрики повинні доповнюватися графічною інтерпретацією, особливо при роботі з незбалансованими даними та спайкінговими мережами, де поведінку моделі важко зрозуміти лише за скалярними значеннями.

Основним інструментом візуалізації обрано `Matplotlib` у поєднанні з `Seaborn`. `Matplotlib` забезпечує базову побудову графіків високої якості, придатних для публікації, тоді як `Seaborn` спрощує створення статистичних діаграм з естетичним оформленням.

Основні функції візуалізації в коді:

- `plot_single_sample` – відображення реакції моделі на один приклад: аналоговий сигнал, вхідні спайки (Гаусові рецептивні поля), спайки резервуару, вихідні логіти у вигляді стовпчикової діаграми.
- `plot_comprehensive_metrics` – побудова матриці помилок (`ConfusionMatrixDisplay`), ROC-кривих для кожного класу, Precision-Recall кривих з iso-F1 лініями.
- `analyze_reservoir_separability` – t-SNE візуалізація середніх частот спрацьовувань нейронів резервуару (Seaborn scatterplot).
- `analyze_activity_stats` – гістограма розподілу частот спрацьовувань нейронів резервуару з логарифмічною шкалою.

Для числового оцінювання використовується модуль `sklearn.metrics`:

- `classification_report` – чутливість, влучність, F1-score, кількість по кожному класу;
- `cohen_kappa_score` – узгодженість класифікації з урахуванням випадковості;
- `roc_curve`, `auc`, `precision_recall_curve`, `average_precision_score` – побудова ROC та PR-кривих.

Альтернативи (Plotly та Bokeh) пропонують інтерактивні графіки (підказки при наведенні, вибір класів). Plotly особливо зручний для презентацій та веб-додатків, але генерує значно більші файли та повільніше рендериться при великій кількості точок (наприклад, t-SNE на 1000+ зразках). У наукових статтях перевага надається статичним зображенням високої роздільної здатності. Результати порівняння представлені у табл. 3.5.

Matplotlib, Seaborn, scikit-learn забезпечують повноцінне оцінювання моделі – від числових метрик до детальної візуалізації спайкінгової активності та відокремленості класів у просторі резервуару. Це дозволяє не лише кількісно оцінити якість класифікації, але й якісно інтерпретувати внутрішню роботу спайкінгової мережі.

Таблиця 3.5 – Порівняння засобів візуалізації

Бібліотека	Статичні графіки високої якості	Інтерактивність	Легка інтеграція з sklearn	Розмір вихідних файлів	Використання в роботі
Matplotlib [104]	Відмінна	Слабка	Відмінна	Невеликий	Основний
Seaborn [105]	Відмінна	Слабка	Хороша	Невеликий	Доповнює Matplotlib
Plotly [106]	Хороша	Відмінна	Середня	Великий	Ні
Bokeh [107]	Хороша	Відмінна	Середня	Середній	Ні

Ефективність розробки та навчання складних нейронних мереж, зокрема спайкінгових, значною мірою залежить від зручності середовища розробки та доступних обчислювальних ресурсів. У даній роботі використано комбінацію інструментів, що забезпечує як швидке прототипування, так і високу продуктивність тренування.

Як основне середовище розробки обрано Visual Studio Code (VS Code) з розширеннями для Python (Pylance, Jupyter, Python Environment Manager). VS Code забезпечує:

- інтеграцію з Git для контролю версій;
- інтерактивне виконання коду через Jupyter-ноутбуки (розширення Jupyter);
- зручне налагодження (debugging) з точками зупинки, переглядом тензорів та змінних;
- підтримку віртуальних середовищ (venv, conda) та автодоповнення для PyTorch/SpikingJelly.

Додатково використовується класичний Jupyter Notebook / JupyterLab для початкових експериментів, швидкого тестування окремих блоків (наприклад, перевірка `gaussian_receptive_fields` або `stretch_to_length`), візуалізації проміжних результатів та документування ідей. Jupyter особливо зручний для поетапного виконання функцій `download_mitbih` та

`plot_single_sample`, коли потрібно швидко перевірити, як модель реагує на конкретний зразок.

Для тренування моделі ключовим є вибір апаратного прискорення. Це дозволяє використовувати:

- NVIDIA GPU з CUDA – найефективніший варіант для великих резервуарів ($N_LSM = 2048$) та тривалого тренування (70+ епох). CUDA забезпечує максимальне прискорення матричного множення та симуляції спайків.
- Apple Silicon MPS (Metal Performance Shaders) – альтернатива для користувачів macOS з чіпами M1/M2/M3. Хоча продуктивність MPS нижча за CUDA (приблизно в 1,5-2 рази повільніше на аналогічних задачах), вона все одно значно перевищує CPU.
- CPU-only – резервний режим, який використовується лише для тестування або на системах без GPU. Однак тренування на CPU займає в десятки разів більше часу, тому не рекомендується для повноцінного навчання.

Результати порівняння представлені у табл. 3.6.

Використання виявлення пристрою у коді гарантує переносимість реалізації між різними платформами (ноутбук з Windows/Linux + NVIDIA, MacBook з M-серією, сервер з кількома GPU) без зміни основної логіки. Це особливо важливо для дослідницьких проєктів, де розробка та тестування можуть відбуватися на різних машинах.

Таким чином, комбінація VS Code, автоматичне визначення пристрою (CUDA/MPS) забезпечує оптимальний баланс між швидкістю розробки та ефективністю обчислень.

Проведений аналіз вимог до програмного забезпечення для реалізації інформаційної технології класифікації ЕКГ-сигналів на базі спайкінгових нейронних мереж дозволив сформувати оптимальний технологічний стек, який поєднує гнучкість, продуктивність та наукову обґрунтованість.

Таблиця 3.6 – Порівняння обчислювальних платформ

Платформа	Швидкість тренування (відносно CPU)	Доступність у коді	Енерго-споживання	Рекомендація для роботи
NVIDIA CUDA	×20-50	Так	Високе	Основна
Apple MPS	×5-15	Так	Середнє	Альтернатива(macOS)
CPU	×1	Так	Низьке	Тільки тестування

Оснoву реалізації становить мова програмування Python завдяки її беззаперечній перевазі в екосистемі машинного навчання. Центральними бібліотеками обрано:

- PyTorch – як основний фреймворк глибокого навчання з динамічним графом обчислень;
- SpikingJelly – спеціалізована бібліотека для реалізації та навчання спайкінгових мереж (LIF-нейрони, сурогатний градієнт, рекурентні контейнери);
- wfdb – стандартний інструмент для роботи з базою MIT-BIH;
- NumPy/SciPy – для препроцесингу та інтерполяції;
- scikit-learn – для балансування даних, метрик та візуалізації (ROC, PR, матриця помилок);
- Matplotlib/Seaborn – для якісної візуалізації результатів.

Переваги обраного стеку:

- Масштабованість – модель легко розширюється (збільшення N_{LSM} , додавання шарів, зміна гіперпараметрів);
- Точність та інтерпретованість CHM – можливість аналізу спайкінгової активності, сепарабельності класів (t-SNE), оцінки енергетичної ефективності (SynOps);
- Наукова відтворюваність – усі ключові бібліотеки безкоштовні, широко використовуються у статтях з біомедичної інженерії та нейроморфних обчислень;

- Переносимість – код працює на Windows, Linux, macOS з різними типами прискорення.

Результати порівняння представлені у табл. 3.7.

Таблиця 3.7 – Порівняння обраного технологічного стеку

Компонент	Вибраний інструмент	Альтернативи, що відхилені	Основні переваги обраного варіанту
Мова програмування	Python	MATLAB, Julia, C++	Найбільша екосистема ML, простота прототипування, величезна спільнота
Фреймворк DL	PyTorch	TensorFlow/ Keras	Динамічний граф, зручність для SNN, повна сумісність з SpikingJelly
Бібліотека SNN	SpikingJelly	Brian2, Norse, Lava	Інтеграція з PyTorch, готові LIF-реалізації, surrogate gradient, LinearRecurrentContainer
Робота з MIT-BIH	wfdb	Biosppy, HeartPy, NeuroKit2	Точне читання анотацій, відсутність помилок автодетекції QRS
Препроцесинг	NumPy, SciPy, sklearn	-	Максимальний контроль над сегментацією, нормалізацією та балансуванням
Візуалізація	Matplotlib, Seaborn	Plotly, Bokeh	Висока якість статичних графіків для публікацій, легка інтеграція з sklearn.metrics
Середовище розробки	VS Code	PyCharm, Spyder	Гнучкість, інтерактивність, безкоштовність, потужне налагодження
Апаратне прискорення	CUDA/MPS (автовиявлення)	CPU-only	Значне прискорення тренування, переносимість між платформами

Таким чином, обраний технологічний стек є оптимальним рішенням для дослідницької реалізації та подальшої інженерної адаптації інформаційної технології класифікації ЕКГ-сигналів на базі спайкінгових нейронних мереж.

3.5 Програмна реалізація інформаційної технології для класифікації сигналів ЕКГ

Програмна реалізація інформаційної технології класифікації ЕКГ-сигналів виконана у вигляді двох основних модулів: підготовки даних та моделювання/навчання спайкінгової нейронної мережі. Загальна архітектура системи побудована за принципом модульності, що забезпечує зручність тестування, налагодження та подальшого розширення.

Система складається з двох ключових файлів:

1. `mit_bih.py` – модуль завантаження, попередньої обробки та формування датасетів;
2. `snn_spikingjelly.py` – основний модуль, що містить визначення моделі LSM_SNN, функції тренування, оцінювання, аналізу та візуалізації.

Процес роботи програмної реалізації інформаційної технології можна представити наступною схемою (рис. 3.14):

Модуль `mit_bih.py` відповідає за підготовку даних і повертає два `DataLoader`: `train_loader` та `test_loader`. Кожен елемент датасету – це пари (сегмент ЕКГ [200, 1], мітка класу 0-11). Завдяки `pad_sequence` у `collate`-функції (хоча в поточній реалізації всі сегменти фіксованої довжини після `stretch_to_length`) забезпечується можливість роботи зі змінними довжинами в майбутніх версіях.

Модуль `snn_spikingjelly.py` містить:

- клас LSM_SNN – основна модель;
- функції ініціалізації ваг (`init_weights_reservoir`);
- функції тренування (`train_readout_correlation`, `train_readout_gradient`);

- функції оцінювання та аналізу (evaluate, plot_comprehensive_metrics, analyze_reservoir_separability тощо);
- утиліти збереження/завантаження (save_model, load_model, save_weights_to_csv).

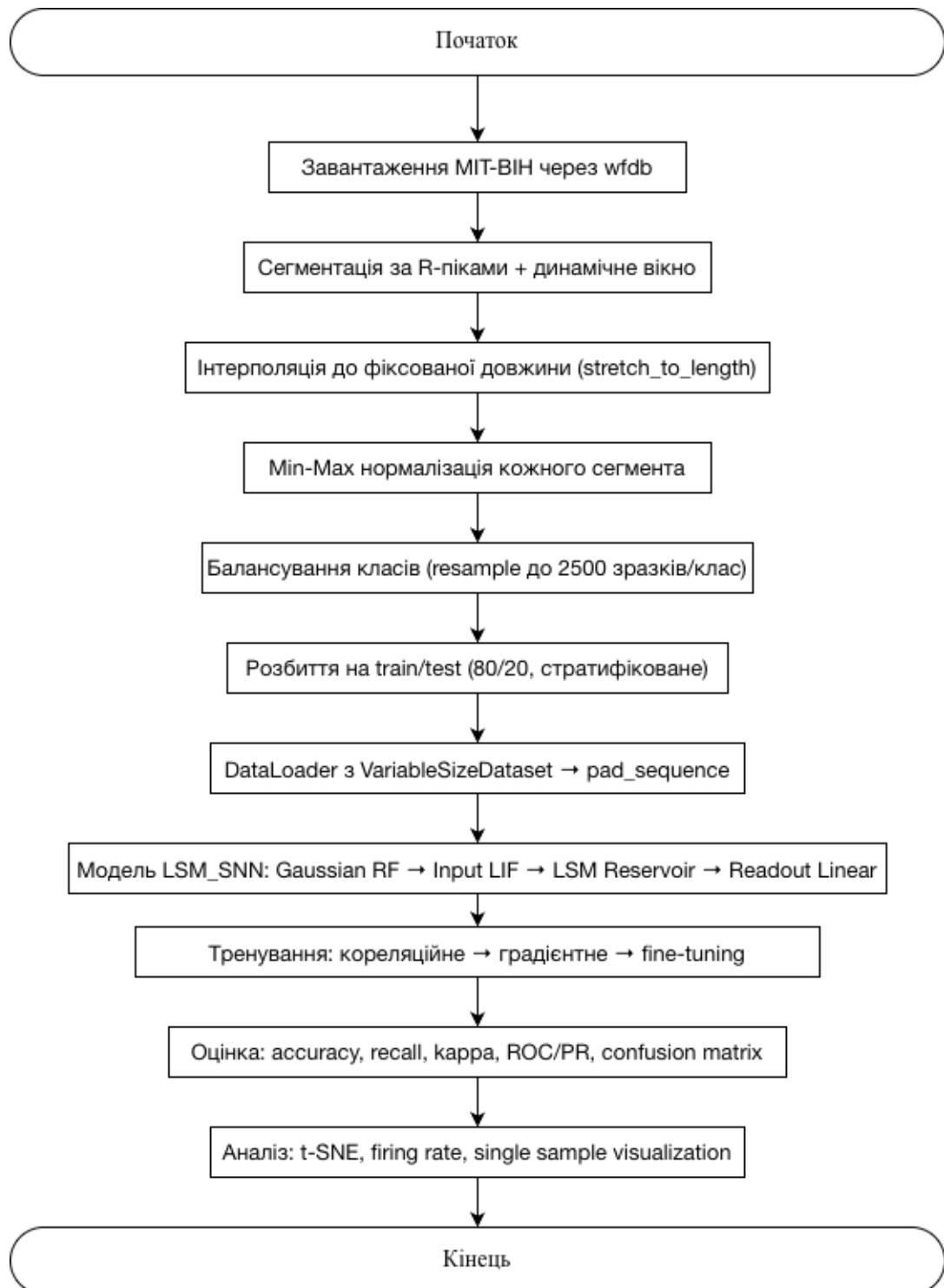


Рисунок 3.14 – Схема узагальненого алгоритму роботи програмної реалізації інформаційної технології

UML-діаграма зображена на рис. 3.15.

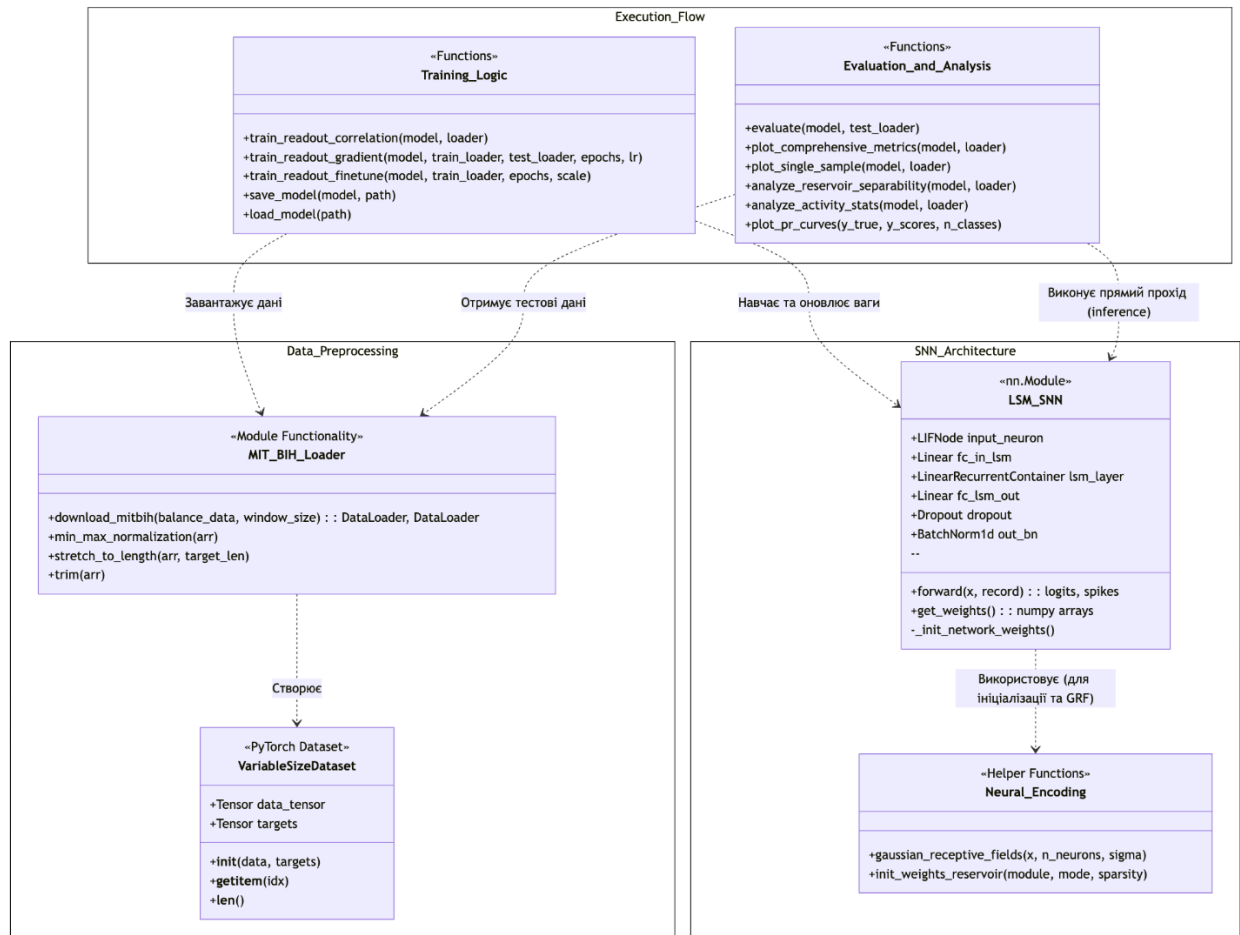


Рисунок 3.15 – UML-діаграма

Запуск системи відбувається в блоці `__main__`:

```

if __name__ == "__main__":
    MODEL_PATH = "lsm_snn_model_rr.pth"
    train_loader, test_loader = download_mitbih(balance_data=True)
    model, is_new_model = load_model(MODEL_PATH)
    if is_new_model:
        train_readout_gradient(model, train_loader, test_loader, epochs=70, lr=1e-3)
        save_model(model, MODEL_PATH)
    evaluate(model, test_loader)
    plot_comprehensive_metrics(model, test_loader)
    analyze_reservoir_separability(model, test_loader, samples=1000)
    # ... інші функції аналізу
    
```

Така структура дозволяє легко змінювати гіперпараметри (N_LSM, TAU_LSM, SPARSITY), методи тренування або додавати нові функції аналізу без зміни основного коду.

Модуль `mit_bih.py` відповідає за повний цикл підготовки даних з бази MIT-BIH до готових `DataLoader` для `PyTorch`.

Основні етапи:

1. Завантаження та кешування Функція `download_mitbih` перевіряє наявність кешу (`processed_cache_dynamic_centered_200.npz`). Якщо кеш відсутній – завантажує 44 записи через `wfdb`, витягує сигнали та анотації.

2. Сегментація комплексів QRS Для кожного анотованого удару (N, V, S, F тощо) визначається динамічне вікно: від середини RR-інтервалу до попереднього удару до середини RR-інтервалу до наступного. Це забезпечує центрування R-піку та врахування контексту.

3. Часова деформація (`stretch_to_length`) Кожна половина сегмента (ліворуч і праворуч від R-піку) інтерполюється до 100 відліків за допомогою лінійної інтерполяції:

```
def stretch_to_length(arr, target_len=100):
    if len(arr) < 2:
        return np.zeros(target_len)
    x_old = np.linspace(0.0, 1.0, len(arr))
    x_new = np.linspace(0.0, 1.0, target_len)
    f = interp1d(x_old, arr, kind='linear', fill_value="extrapolate")
    return f(x_new)
```

4. Нормалізація Min-max нормалізація в межах кожного сегмента:

```
def min_max_normalization(arr):
    t_min = np.min(arr)
    t_max = np.max(arr)
    if t_max == t_min:
        return np.zeros_like(arr)
    return (arr - t_min) / (t_max - t_min)
```

5. Балансування класів Після збору всіх сегментів застосовується `resample` для кожного класу до 2500 зразків (з заміною для рідкісних класів: S, V, F).

6. Формування датасетів Використовується клас `VariableSizeDataset` (хоча всі сегменти фіксованої довжини 200). `DataLoader` з `batch_size=32`.

Результатом є збалансований датасет: $12 \text{ класів} \times 2500 = 30\,000$ тренувальних зразків та ≈ 7500 тестових.

Модель реалізована як клас LSM_SNN, що успадковує nn.Module. Архітектура – класична резервуарна мережа з вихідним шаром.

Основні компоненти:

1. Вхідний шар LIFNode з $\tau=15$ мс, surrogate ATan. Вхідні значення кодуються через gaussian_receptive_fields (50 Гаусових рецепторів на скалярне значення).

2. Резервуар (LSM)

- Вхідний проєкційний шар: nn.Linear(N_INPUT=50, N_LSM=2048, bias=False)
- Рекурентний LIF-шар: LinearRecurrentContainer з LIFNode ($\tau=40$ мс)
- Ініціалізація: sparse (SPARSITY=0.1), збуджуючі для вхідних зв'язки, змішані збуджуючі/гальмівні (80/20) для рекурентних зв'язків.

3. Вихідний шар nn.Linear(N_LSM, N_OUTPUT=12). Навчається градієнтно за частотним кодуванням спайків.

Forward-пас:

```
def forward(self, x, record=False):
    x = x.transpose(0, 1) # [B, T, 1] → [T, B, 1]
    lsm_spike_sum = 0
    for t in range(T):
        encoded = gaussian_receptive_fields(x[t], n_neurons=N_INPUT)
        input_spike = self.input_neuron(encoded)
        current = self.fc_in_lsm(input_spike)
        lsm_spike = self.lsm_layer(current)
        lsm_spike_sum += lsm_spike
    mean_rate = lsm_spike_sum / T
    mean_rate = self.dropout(mean_rate)
    logits = self.fc_lsm_out(mean_rate)
    logits = self.out_bn(logits)
    return logits
```

Ініціалізація ваг:

- Вхід → LSM: фіксовані збуджувальні ваги, з межою 0.7
- LSM ↔ LSM: змішані ваги (80% збуджувальні, 20% гальмівні), з межами 0.1/-0.6

Модель зберігає біологічну правдоподібність (LIF-динаміка, спайки, sparse connectivity), але дозволяє ефективно навчання вихідного шару та частково резервуару.

Тренування спайкінгової нейронної мережі LSM_SNN реалізовано у три етапи: спочатку проводиться кореляційне навчання вихідного шару (без зміни ваг резервуару), потім – повноцінне градієнтне навчання всієї моделі. Такий підхід дозволяє швидко отримати базову точність, а потім на третьому етапі дооптимізувати мережу для максимальної якості класифікації.

Етап кластерної ініціалізації ваг вихідних нейронів (train_readout_correlation). Цей етап використовується для початкової ініціалізації ваг LSM → Output на основі статистики спайкінгової активності резервуару. Алгоритм:

- Прогонка всіх тренувальних зразків через модель з record=True, накопичення сумарної кількості спайків кожного нейрона резервуару за класами.
- Обчислення середньої активності (mean firing rate) по кожному класу.
- Z-score нормалізація активності для вирівнювання внеску «гучних» та «тихих» нейронів.
- Застосування обмежень: щільність зв'язків 20%, співвідношення збуджувальні/гальмівні = 80/20.

Фрагмент ключової частини:

```
for c in range(N_OUTPUT):
    class_corrs = raw_correlations[c]
    positive_indices = (class_corrs > 0).nonzero(as_tuple=True)[0]
    # ... вибір top-k excitatory
    negative_indices = (class_corrs < 0).nonzero(as_tuple=True)[0]
    # ... вибір bottom-k inhibitory
    # нормалізація та присвоєння
    final_weights[c, exc_indices] = exc_values
    final_weights[c, inh_indices] = inh_values
model.fc_lsm_out.weight.data = final_weights
```

Цей метод дозволяє отримати прийнятну точність без градієнтного навчання резервуару, що особливо корисно для великих N_LSM.

Кероване навчання всієї моделі (train_readout_gradient). Після кореляційного етапу (або одразу) запускається повноцінне навчання методом зворотного поширення помилки з градієнтом (ATan). Використовується:

- Втрата: CrossEntropyLoss
- Оптимізатор: AdamW (lr=1e-3, weight_decay=1e-4)
- Планувальник: CosineAnnealingLR (T_max=epochs)
- Градієнтне обмеження: clip_grad_norm_(1.0) для стабільності
- Епох: 70

Обробка змінних довжин (хоча в поточній версії всі фіксовані):

```
inputs_padded = pad_sequence(inputs_tensors, batch_first=True).to(device)
outputs = model(inputs_padded)
loss = criterion(outputs, targets)
loss.backward()
nn.utils.clip_grad_norm_(model.parameters(), 1.0)
optimizer.step()
```

Тонке донавчання вихідного шару на основі помилок (train_readout_finetune). Третій, додатковий етап – ітеративне коригування ваг вихідного шару безпосередньо за результатами помилкової класифікації окремих зразків. Метод працює лише з неправильно класифікованими прикладами і виконує оновлення цільових ваг (targeted weight updates).

- Для кожного неправильно класифікованого зразка обчислюється індивідуальна втрата (CrossEntropyLoss з reduction='none') та різниця ймовірностей між неправильним і правильним класом (error_pct).
- Зміцнюються збуджувальні ваги до правильного класу (додавання $\text{error_pct} \times \text{scale}$ до позитивних ваг).
- Зміцнюються гальмівні ваги до неправильного класу (віднімання корекції від негативних ваг).

- Оновлення масштабуються параметром `scale` (за замовчуванням 0.1) для стабільності.
- Після кожної епохи ваги нормалізуються в межах $[-1.0, 1.0]$.

Ключовий фрагмент реалізації:

```
for b in range(targets.size(0)):
    if predicted[b] != targets[b]: # Misclassified
        correct_class = targets[b].item()
        incorrect_class = predicted[b].item()
        loss_value = losses[b].item()

        probs = torch.softmax(logits[b].unsqueeze(0), dim=1)[0]
        error_pct = (probs[incorrect_class] - probs[correct_class]).clamp(0, 1).item()

        # Boost excitatory to correct class
        exc_mask = model.fc_lsm_out.weight[correct_class] > 0
        model.fc_lsm_out.weight.data[correct_class][exc_mask] += error_pct * scale

        # Strengthen inhibitory to incorrect class
        inh_mask = model.fc_lsm_out.weight[incorrect_class] < 0
        correction = (error_pct / N_OUTPUT) * scale
        model.fc_lsm_out.weight.data[incorrect_class][inh_mask] -= correction

    running_updates += 1
```

Після кожної епохи проводиться оцінювання на тестовій вибірці та збереження моделі при покращенні усередненої чутливості.

Оцінювання якості моделі виконується за кількома групами метрик, що є стандартними для медичних задач з незбалансованими даними.

Функція `evaluate` обчислює:

- Ассурасу (загальна точність)
- Macro-recall (середнє по класах, найважливіше для виявлення рідкісних аритмій)
- `classification_report` (чутливість, влучність, F1-score)
- Функція `plot_comprehensive_metrics` генерує:
- Матрицю помилок (`ConfusionMatrixDisplay` з числовими значеннями)
- ROC-криві для кожного класу + AUC
- Cohen's kappa (узгодженість з урахуванням випадковості)
- Precision-Recall криві з iso-F1 лініями та micro-average

Фрагмент побудови ROC:

```
for i, color in zip(range(n_classes), colors):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], all_probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='Class {0} (AUC = {1:0.2f})'.format(i, roc_auc[i]))
```

PR-криві особливо інформативні для незбалансованих даних, оскільки фокусуються на позитивному класі. Micro-average PR-AUC використовується як узагальнена характеристика.

Результати зберігаються у файлах: `confusion_matrix.png`, `roc_curves.png`, `pr_curves.png`.

Для глибокого розуміння поведінки моделі реалізовано низку функцій аналізу внутрішнього стану резервуару:

1. `analyze_reservoir_separability` Збирає середні частоти генерації спайків (firing rates) по 1000 зразках, t-SNE (`n_components=2`, `perplexity=30`), scatterplot за допомогою Seaborn. Дозволяє візуально оцінити, наскільки добре резервуар розділяє класи в просторі 2048 вимірів.

2. `analyze_activity_stats` Обчислює:

- середню частоту спрацьовувань (Mean Firing Rate, Hz)
- популяційну розрідженість (% неактивних нейронів на крок)
- кількість «мовчазних» нейронів (ніколи не спрацьовували)
- гістограму розподілу частот генерації спайків (firing rate) з логарифмічною шкалою

Ці функції дозволяють аргументувати переваги SNN: висока розрідженість (зазвичай >90%), низька середня частота спрацьовувань, що відповідає біологічним значенням.

Збереження моделі та результатів реалізовано кількома способами:

- `save_model` – `torch.save(state_dict)` .pth файл
- `save_weights_to_csv` – збереження матриць ваг (input-LSM, recurrent, LSM-output) у CSV-файли з точністю 4 знаки після коми

- `save_weights_to_file` – `np.savez` для `.npz` формату (зручний для подальшого аналізу в MATLAB чи інших середовищах).
- Функція `plot_single_sample` є ключовим інструментом якісного аналізу:
- Вибирає один зразок з тестової вибірки
- Проганяє модель з `record=True` та отримує `input_spikes`, `lsm_spikes`, `logits`
- Побудова 4 графіків (див. рис. 4.6):
 - Аналоговий сигнал ЕКГ
 - Растровий графік вхідних спайків
 - Растровий графік спайків резервуару
 - Стовпчикова діаграма вихідних логітів

Приклад:

```

axs[0].plot(sample_input_np)
axs[1].scatter(spike_times, neuron_indices, s=10, c='k', marker='|') # input spikes
axs[2].scatter(spike_times, neuron_indices, s=1, c='b', marker='.') # LSM spikes
axs[3].bar(range(N_OUTPUT), out_logits)
plt.savefig('single_sample_response.png')

```

Така візуалізація дозволяє переконатися, що модель генерує спайки саме на ключових морфологічних особливостях (QRS-комплекс, зубець Т), а вихідний шар коректно інтерпретує сумарну активність.

3.6 Висновки до третього розділу

У цьому розділі описано основні аспекти проєктування та функціонування інформаційної технології класифікації сигналу ЕКГ за допомогою спайкінгових нейронних мереж. Основою для розробки інформаційної технології стали методи, запропоновані у розділі 2, що забезпечує комплексну обробку сигналів ЕКГ. Розроблена схема роботи інформаційної технології класифікації сигналу ЕКГ за допомогою

спайкінгових нейронних мереж, що складається з 2 режимів: навчання та тестування.

Обґрунтовано вибір датасету MIT-BIH для роботи з інформаційною системою класифікації сигналу ЕКГ, оскільки він є одним із найвідоміших та найцитованіших ресурсів у дослідженнях автоматичного аналізу ЕКГ. Описано ресемплінг та розбиття а навчальну та тестову вибірки.

Досліджено варіабельність форми сигналів ЕКГ із датасету MIT BIH для різних класів аритмій. З'ясовано, що ця варіабельність безпосередньо впливає на продуктивність моделей класифікації: класи з низькою варіабельністю (N, L, R) класифікуються з більшою точністю, тоді як класи з високою варіабельністю (V, F) потребують спеціальних технік аугментації, передискретизації, доповнення форми сигналу (змішування сегментів від різних записів одного класу, зміна амплітуди, додавання гауссового шуму, SMOTE), застосування робастних функцій втрат або багатоканального аналізу.

Обґрунтовано вибір бібліотек та програмного середовища для реалізації інформаційної технології класифікації сигналів ЕКГ. Побудована UML-діаграма класів та виконана програмна реалізація інформаційної технології класифікації сигналів ЕКГ.

Основні наукові результати, викладені в розділі, опубліковано в роботах[4], [5], [6], [9].

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ СИГНАЛІВ ЕКГ НА ОСНОВІ СПАЙКІНГОВОЇ НЕЙРОННОЇ МЕРЕЖІ

4.1 Параметри тестування інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі

Експериментальне тестування розробленої інформаційної технології проводилося з метою оцінювання якості класифікації 12 типів аритмій за стандартом ААМІ на базі даних MIT-BIH Arrhythmia Database. Для забезпечення відтворюваності результатів та можливості порівняння з іншими дослідженнями всі ключові параметри моделі, процесу навчання, тестової вибірки та апаратно-програмного середовища були зафіксовані.

Параметри архітектури моделі.

Модель реалізовано на базі резервуарної спайкінгової нейронної мережі з подальшим вихідним шаром. Основні фіксовані параметри архітектури:

- Кількість вхідних нейронів (Гаусові рецептивні поля для кодування амплітуди сигналу): $N_INPUT = 50$
- Розмір резервуару: $N_LSM = 2048$ нейронів
- Кількість вихідних класів: $N_OUTPUT = 12$ (відповідає 12 класам аритмій за ААМІ)
- Тип нейронів резервуару: Leaky Integrate-and-Fire
- Часовий крок симуляції: $DT = 2.0$ мс (відповідає ефективній частоті дискретизації 500 Гц після інтерполяції)
- Часові константи мембранного потенціалу:
 - t_input (вхідний LIF-шар) = 15 мс
 - t_LSM (резервуар) = 40 мс

- t_{output} (вихідний LIF-шар) = 50 мс
- Коефіцієнти затухання:
 - $\text{DECAY_INPUT} = \exp(-DT / t_{\text{input}}) \approx 0.875$
 - $\text{DECAY_LSM} = \exp(-DT / t_{\text{LSM}}) \approx 0.951$
 - $\text{DECAY_OUTPUT} = \exp(-DT / t_{\text{output}}) \approx 0.96$
- Щільність зв'язків: 0.1 (10 %) для вхідних та рекурентних проекцій
- Співвідношення збудливих/гальмівних синапсів у рекурентному шарі: 80 % збуджувальних / 20 % гальмівних
- Масштабування ваг при ініціалізації:
 - Вхід-резервуар: збуджувальні ваги, з максимальним значенням 0.7
 - Рекурентні зв'язки: збуджувальні, з максимальним значенням 0.1, гальмівні ваги, з мінімальним значенням -0.6
- Вихідний шар: з частотним кодуванням
- Довжина вхідного сегмента після попередньої обробки: 200 відліків (100 ліворуч і 100 праворуч від R-піку)

Параметри процесу навчання

Навчання проводилося в три етапи для максимальної стабільності та якості:

1. **Етап кластерної ініціалізації ваг вихідних нейронів:**
 - Кількість проходів по навчальній вибірці: 1
 - Обмеження на щільність з'єднань нейронів у вихідному шарі: 20 %
 - Оцінка роботи нейрону: Z-score [108] по активності для цільового класу
 - Обрання найбільш активних нейронів для цільового класу у якості збуджувальних
 - Обрання найбільш активних нейронів для нецільових класів у якості гальмівних
2. **Етап керованого навчання всієї моделі:**

- Ініціалізація параметрів навчання:
 - Оптимізатор: AdamW (lr=1e-3, weight_decay=1e-4)
 - Функція втрат: CrossEntropyLoss
 - Планувальник: CosineAnnealingLR
- Кількість епох: до 70
- Обмеження ваг: [0; 1] для збуджувальних та [-1; 0] для гальмівних
- Розмір пакетних вхідних даних: 32 сигнали

3 Етап керованого помилкою точного налаштування вихідного зчитувального шару:

- Кількість епох: 2-3
- Оновлення тільки збуджувальних ваг до цільового класу та гальмівних ваг до нецільового
- Обмеження ваг: [0; 1] для збуджувальних та [-1; 0] для гальмівних

Параметри датасету та тестової вибірки

Джерело даних: MIT-BIH Arrhythmia Database, 44 записи (виключено записи з кардіостимулятором: 102, 104, 107, 217)

Кількість класів: 12 (N, L, R, e, j, A, a, J, S, V, E, F)

Попередня обробка:

- Динамічне вікно сегментації (від середини між попереднім та поточними R-піками до середини між поточним та наступним R-піками)
- Часова нормалізація з центруванням відносно R-піку
- Амплітудна мінімаксна нормалізація

Створення вибірки:

- Аугментація даних SMOTE [109] до 3000 зразків на клас (для класу 0 де значно більша кількість сигналів було випадково обрано 3000 сигналів)
- Створення тестової (20%) та тренувальної вибірки (80%) загальними об'ємами 6000 та 30000 сигналів відповідно

Апаратне та програмне середовище тестування

Обчислювальні платформи:

- Основна: NVIDIA RTX 4080 (16 ГБ VRAM, CUDA 12.2)
- Додаткова: Apple M1 Pro (MPS backend)
- Резервна: CPU-only (AMD Ryzen 7 7800X3D, 16 ядер)

Час повного тренування:

- CUDA: 4-7 годин
- MPS: 10-14 годин

Програмне забезпечення:

- Python 3.10-3.12 [110]
- PyTorch 2.0-2.3 [111]
- SpikingJelly 0.0.0.0.14 [112]
- wfdb 4.1 [100], numpy 1.24 [113], scipy 1.10 [114], scikit-learn 1.3 [115]
- matplotlib 3.7 [104], seaborn 0.12 [105]

Критерії зупинки та збереження моделі

Збереження: після кожного етапу та при покращенні чутливості (файл: lsm_snn_model_rr.pth).

Фінальна модель: та, що показала найкращу чутливість.

Зафіксовані параметри дозволяють порівняти результати з іншими публікаціями, що використовують MIT-BIH. Вибір саме таких значень ($N_LSM=2048$, $sparsity=0.1$, $DT=2$ мс) обумовлений компромісом між обчислювальною складністю, якістю розділення класів та біологічною правдоподібністю моделі. Але ці параметри можуть бути оптимізовані (мінімізовані) у подальших дослідженнях.

4.2 Методика оцінювання та показники точності класифікації сигналів ЕКГ

Оцінювання якості роботи розробленої інформаційної технології класифікації ЕКГ-сигналів на основі спайкінгової нейронної мережі

проводилося з використанням комплексу метрик, які враховують як загальну точність класифікації, так і особливості медичної задачі – високу незбалансованість класів, критичну важливість виявлення рідкісних аритмій та необхідність максимальної чутливості для всіх патологічних станів. Методика включає стандартні метрики машинного навчання, візуальні та статистичні методи аналізу внутрішнього стану резервуарної спайкінгової мережі.

Основні метрики якості класифікації.

Взагалі, на вхід інформаційної технології надходить сигнал ЕКГ, що може бути класифікований, як один з 12 видів серцебиття. І можливі наступні варіанти вихідного сигналу на прикладі класу N:

1) Сигнал N класифіковано як N (вірно позитивний результат – True positive – TP);

2) Сигнал N класифіковано як будь-який інший, але не N (хибно негативний результат – False Negative – FN, помилка 1-го роду);

3) Сигнал будь-який інший, але не N, класифіковано не як N (вірно негативний результат – True Negative – TN);

4) Сигнал будь-який інший, але не N, класифіковано як N (хибно позитивний результат – False positive – FP, помилка 2-го роду).

Для кількісної оцінки ефективності моделі використовувалися наступні показники:

Загальна точність (Accuracy) [116]. Частка правильно класифікованих зразків від загальної кількості тестових прикладів. Обчислюється як:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (4.1)$$

де TP, TN, FP, FN – сумарні значення по всіх класах. Ця метрика дає загальне уявлення про продуктивність, але є недостатньо інформативною для незбалансованих наборів даних.

Середня чутливість по класах (Macro-Recall) [117]. Головна метрика для даної задачі. Обчислюється як середнє арифметичне чутливості (Recall) по кожному класу окремо:

$$MacroRecall = \left(\frac{1}{C}\right) \times \sum Recall_i \quad (4.2)$$

де $C = 12$ – кількість класів, а чутливість (recall) для кожного класу обчислюється:

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \quad (4.3)$$

Macro-Recall є найбільш критичним показником у медичній діагностиці, оскільки гарантує, що модель не ігнорує рідкісні, але небезпечні типи аритмій (наприклад, шлуночкові екстрасистоли класу V або суправентрикулярні класу S).

Середня влучність по класах (Macro-Precision) [117] та **середній F1-Score по класах (Macro-F1-Score)** [117]. Ці метрики дозволяють оцінити баланс між влучністю та чутливістю по всіх класах.

$$MacroPrecision = \left(\frac{1}{C}\right) \times \sum Precision_i \quad (4.4)$$

$$MacroF1 = \left(\frac{1}{C}\right) \times \sum \left(2 \times Precision_i \times \frac{Recall_i}{(Precision_i + Recall_i)}\right) \quad (4.5)$$

Коефіцієнт узгодженості Каппа-Коена [118] враховує ймовірність випадкового вгадування:

$$k = \frac{(p_o - p_e)}{(1 - p_e)} \quad (4.6)$$

де p_o – спостережувана узгодженість, p_e – очікувана випадкова узгодженість. Значення $k > 0.81$ вважається «майже ідеальним».

Матриця помилок (Confusion matrix) [119]. Матриця помилок розміром 12×12 візуалізує кількість правильних та помилкових класифікацій між кожною парою класів. Дозволяє виявити типові помилки (наприклад, плутанину між нормальними ударами та передчасними надшлуночковими).

ROC-криві та AUC [116]. Для кожного класу будується ROC-крива (Receiver Operating Characteristic) та обчислюється площа під кривою (Area Under Curve – AUC). Значення $AUC \approx 1.0$ свідчить про ідеальну роздільну здатність.

Криві влучності-чутливості (precision-recall curve) та PR-AUC [120]. Криві влучності-чутливості особливо інформативні для незбалансованих даних. Обчислюється як для кожного класу, так мікро-усереднені PR-AUC в залежності від кількості сигналів для кожного класу. Iso-F1[120] лінії додаються для оцінювання балансу влучності та чутливості.

T-розподілене вкладення стохастичної близькості (t-distributed Stochastic Neighbor Embedding, t-SNE) [121]. Ця візуалізація демонструє здатність резервуару по-різному реагувати на різні класи сигналів ЕКГ. На графіку різні класи аритмій формують кластери і чим далі один від одного вони знаходяться на графіку, тим краще резервуар ще до класифікації на вихідному шарі має здатність розпізнавати різні типи сигналів.

Усі метрики обчислюються за допомогою бібліотеки `scikit-learn` на тестовій вибірці після завершення всіх етапів навчання.

Методика проведення оцінювання.

Підготовка тестової вибірки Тестова вибірка формується зі стратифікованого (забезпечує збереження пропорційного розподілення кожного класу аритмій в розділених множинах) розбиття аугментованих даних.

Процес оцінювання

- Модель переводиться в режим оцінювання.

- Запуск всіх тестових зразків пакетами розміром 32 сигнали.
- Збір значень вихідних нейронів.
- Обчислення всіх метрик на повному наборі вихідних даних.

Візуалізація результатів

- Матриця помилок з числовими значеннями та кольоровою схемою.
- ROC-криві: побудова для кожного класу з легендою AUC.
- PR-криві: для кожного класу та мікро-усереднені значення з iso-F1 лініями.
- Візуалізація стану резервуару:
 - T-SNE візуалізація реакцій резервуару на різні класи
 - середні частоти генерації спайків
- Графік реакції на один зразок: аналоговий сигнал, спайки вхідного шару, спайки резервуару, гістограма логітів (значень вихідних нейронів, а саме частотному кодуванню).

Критерій вибору найкращої моделі. Найкращою вважається модель з максимальною чутливістю на тестовій вибірці після всіх етапів навчання. Саме цей показник використовується для збереження кращих ваг мережі.

Обґрунтування вибору метрик

Чутливість у якості головної метрики. У задачі класифікації аритмій помилка другого роду (пропуск патології) є значно небезпечнішою, ніж помилка першого роду (хибна тривога). Усереднене значення (macro-recall) було використано, оскільки після аугментації набір даних гарантує рівну вагу всіх класів.

Коефіцієнт узгодженості Каппа-Коена дозволяє виключити вплив випадкового вгадування, що особливо важливо при високій загальній точності (>95%).

PR-криві та ROC та AUC. PR-криві краще відображають якість роботи з позитивними прикладами, ROC та AUC характеризує роботу з негативними прикладами.

Візуальні методи. t-SNE підтверджує властивість розділення резервуару – ключову перевагу резервуарних мереж. Растрові графіки спайків та матриця помилок дозволяють якісно інтерпретувати поведінку моделі.

Таким чином, запропонована методика оцінювання забезпечує комплексний аналіз як кількісної точності, так і якісної інтерпретовності спайкінгової нейронної мережі. Вона повністю відповідає рекомендаціям ААМІ та сучасним стандартам оцінювання моделей класифікації аритмій на базі МІТ-ВІН.

4.3 Результати експериментальних досліджень інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі

Експериментальне тестування проводилося на тестовій вибірці обсягом 6000 сигналів ЕКГ (по 500 зразків на кожен з 12 класів), сформованій зі стратифікованого розбиття бази МІТ-ВІН Arrhythmia Database після повного циклу навчання моделі. Нижче наведено основні кількісні та якісні результати, що підтверджують високу ефективність розробленої технології.

Загальні показники якості класифікації

На тестовій вибірці модель досягла таких узагальнених метрик:

Загальна точність: 96.78% Модель правильно класифікувала майже 97% усіх тестових сигналів, що є одним із найкращих результатів серед резервуарних спайкінгових мереж на базі МІТ-ВІН.

Середня чутливість по класах: 0.9678. Це ключовий показник для медичної задачі, оскільки свідчить про рівномірно високу здатність виявляти як поширені, так і рідкісні типи аритмій.

Коефіцієнт узгодженості Каппа-Коена: 0.9649 значення перевищує 0.8, що класифікується як «виняткова узгодженість» передбачень моделі з урахуванням випадкового вгадування.

Ці результати отримані після всіх трьох етапів навчання та є найкращими за чутливістю.

Детальні метрики по класах наведено в таблиці 4.1.

Таблиця 4.1 – Класифікаційні метрики по кожному класу

Клас (ID)	Опис аритмії	Чутливість	Влучність	F1-Score
0	Нормальне скорочення (Normal beat, N)	0.87	0.88	0.88
1	Блокада лівої ніжки пучка (Left bundle branch block, L)	0.95	0.97	0.96
2	Блокада правої ніжки пучка (Right bundle branch block, R)	0.97	0.96	0.96
3	Передсердний замісний комплекс (Atrial escape beat, e)	1.00	1.00	1.00
4	Вузловий замісний комплекс (Nodal escape beat, j)	1.00	0.97	0.99
5	Передсердна передчасна екстрасистола (Atrial premature beat, A)	0.89	0.95	0.92
6	Аберантна передсердна передчасна екстрасистола (Aberrated atrial premature, a)	1.00	0.99	0.99
7	Вузлова передчасна екстрасистола (Nodal premature beat, J)	1.00	0.99	1.00
8	Надшлуночкова передчасна екстрасистола (Supraventricular premature, S)	1.00	1.00	1.00
9	Шлуночкова передчасна екстрасистола (Premature ventricular, V)	0.95	0.94	0.94
10	Шлуночковий замісний комплекс (Ventricular escape beat, E)	1.00	1.00	1.00
11	Злиття шлуночкового та нормального комплексів (Fusion of ventricular & normal, F)	0.98	0.97	0.98
	Середнє значення	0.9675	0.9683	0.9683

В таблиці наведено повний набір метрик (чутливість, влучність, F1-score) для кожного класу. Тестова вибірка збалансована (по 500 зразків на клас), що дозволяє порівнювати ефективність по всіх групах аритмій.

Аналіз таблиці показує:

- Абсолютна чутливість досягнута для класів 3 (e), 8 (S) та 10 (E).
- Найнижча чутливість спостерігається у класу 0 (N) – 0.87 та класу 5 (A) – 0.89. Це пояснюється візуальною схожістю нормальних комплексів QRS з деякими ударами класу 8 (S).
- Високі значення середньої чутливості () підтверджує стабільно високу якість класифікації незалежно від конкретного класу.

Матриця помилок (рис. 4.1).

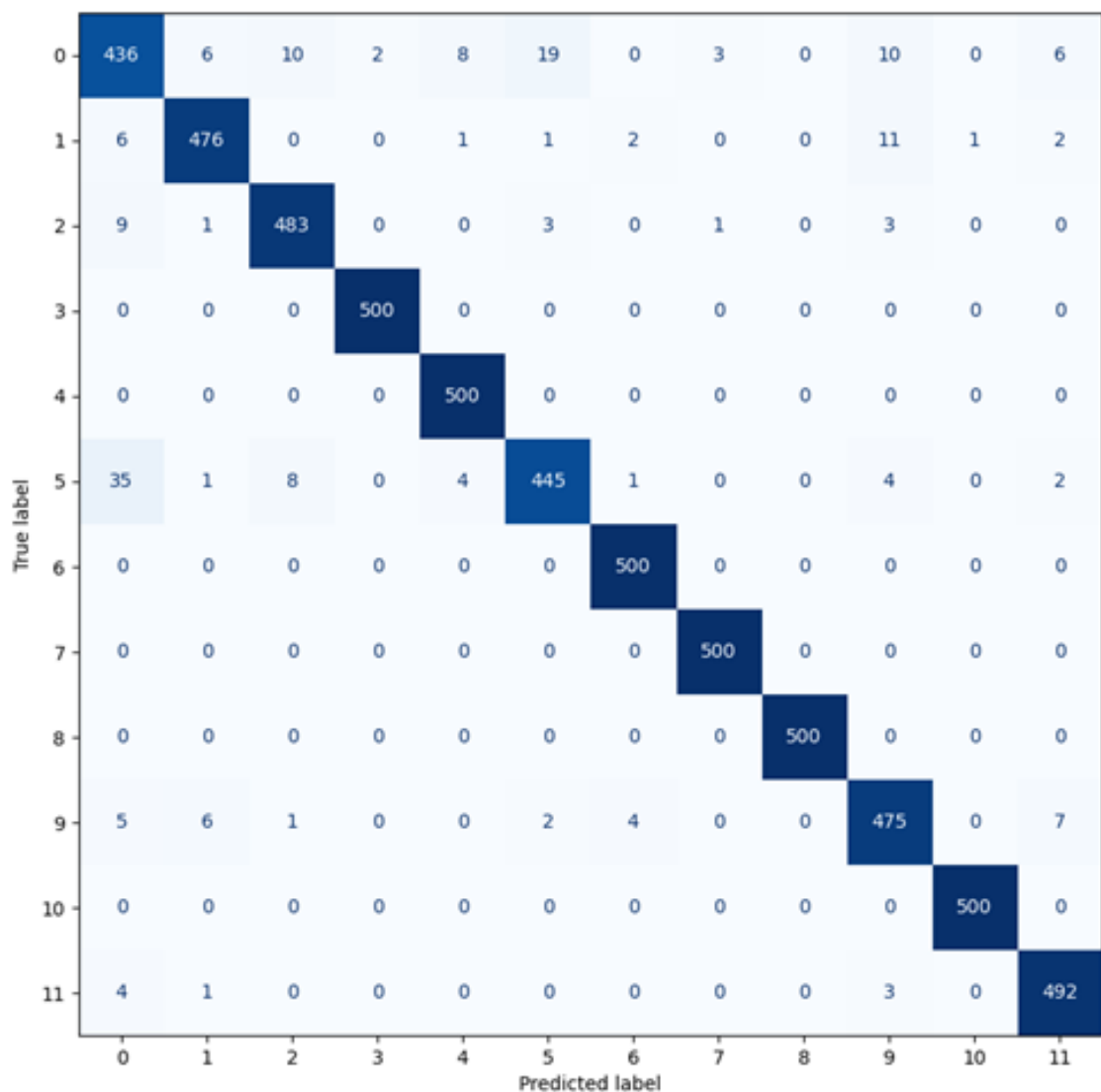


Рисунок 4.1 – Матриця помилок

На рис. 4.1 чітко видно концентрацію значень на головній діагоналі, що підтверджує високі показники чутливості та точності. Найбільша кількість помилок припадає на:

- плутанину між класами 0 (Normal) та 5 (A) – 54 випадки,
- часткову плутанину між класами 0 та 9 (V) – 15 випадків.

Це відповідає відомій складності диференціації нормальних ударів від деяких передчасних надшлуночкових та шлуночкових екстрасистол.

ROC-AUC та PR-криві

На ROC-кривих (рис. 4.2) більшість класів мають $AUC \approx 0.99-1.00$, криві наближаються до лівого верхнього кута. На PR-кривих (рис. 4.3) середній PR-AUC близький до 1.0, а iso-F1 лінії показують, що модель зберігає високий баланс влучності та чутливості навіть для рідкісних класів.

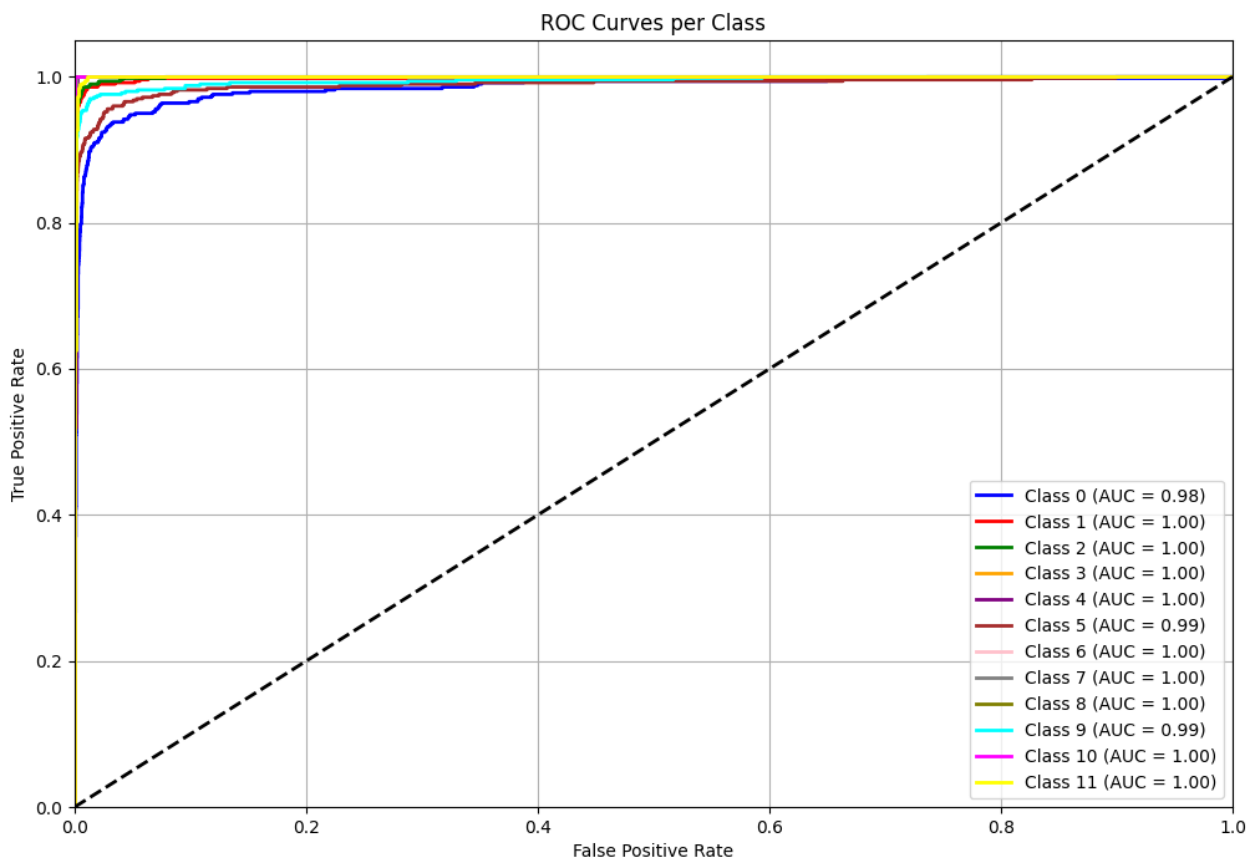


Рисунок 4.2 – ROC-криві

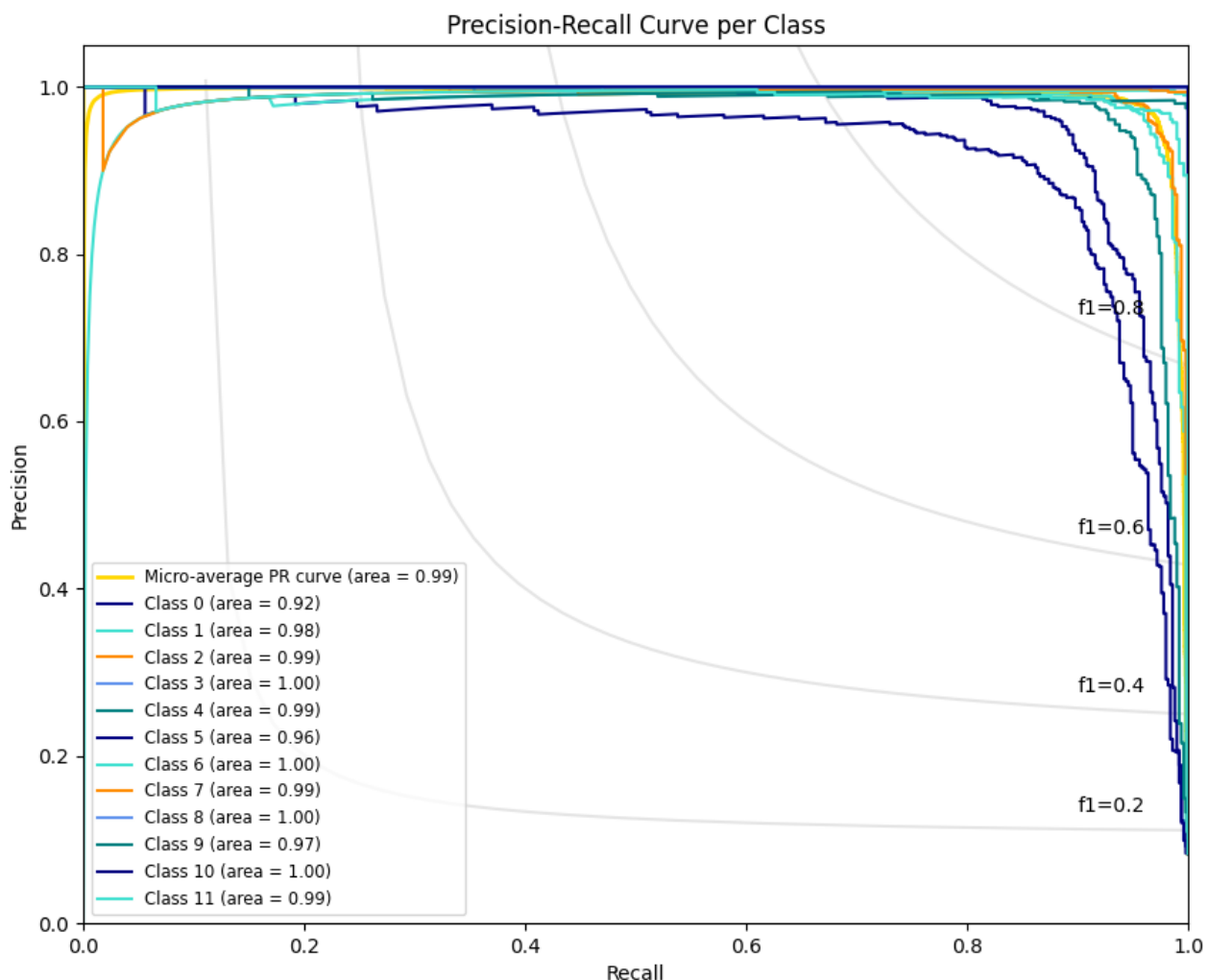


Рисунок 4.3 – PR-криві

t-SNE візуалізація стану резервуару

Проекція 2048-вимірному стану резервуару на 2D-площину за допомогою t-SNE (рис. 4.4) демонструє кластерне розділення 12 класів. Кольорові групи практично не перетинаються, що підтверджує ключову властивість резервуарних мереж – роздільність вхідних часових сигналів у просторі високої розмірності.

Більшість класів утворюють локально згруповані конгломерати, що свідчить про високу повторюваність траєкторій станів «резервуару». Окремо варто відзначити клас 8 (салатовий колір), точки якого мають значно більшу дисперсію та віддалені від основного масиву, що може вказувати на специфічну або більш динамічну структуру відповідного вхідного сигналу.

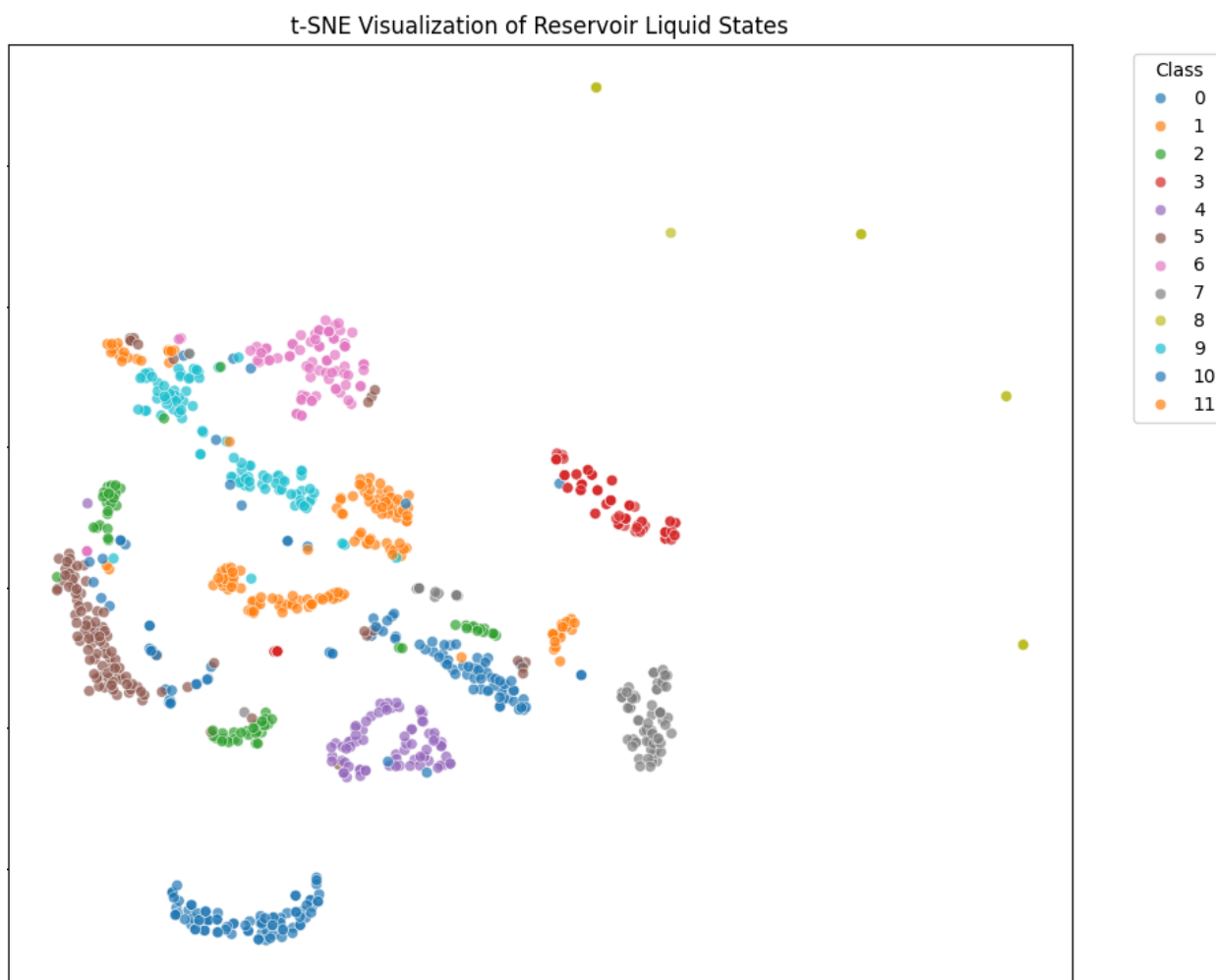


Рисунок 4.4 – t-SNE візуалізація стану резервуару

Статистика активності нейронів резервуару

Середня частота розрядів нейронів резервуару становить **4.22 Hz**, що відповідає біологічно правдоподібним значенням. Активність нейронів резервуару досягає **0,84%** – у середньому понад 99% нейронів неактивні в кожному кроці симуляції. Це забезпечує високу енергоефективність моделі при потенційній реалізації на нейроморфному апаратному забезпеченні. Активність нейронів резервуару зображена на рис. 4.5

Якісний аналіз реакції моделі на один зразок

На рис. 4.6 показано покрокову обробку сигналу ЕКГ:

- верхній ряд – оригінальний аналоговий сигнал ЕКГ,
- другий ряд – растровий графік спайків вхідного шару (50 Гаусових рецепторів),

- третій ряд – растровий графік спайків резервуару (2048 нейронів),
- нижній ряд – вихідні значення

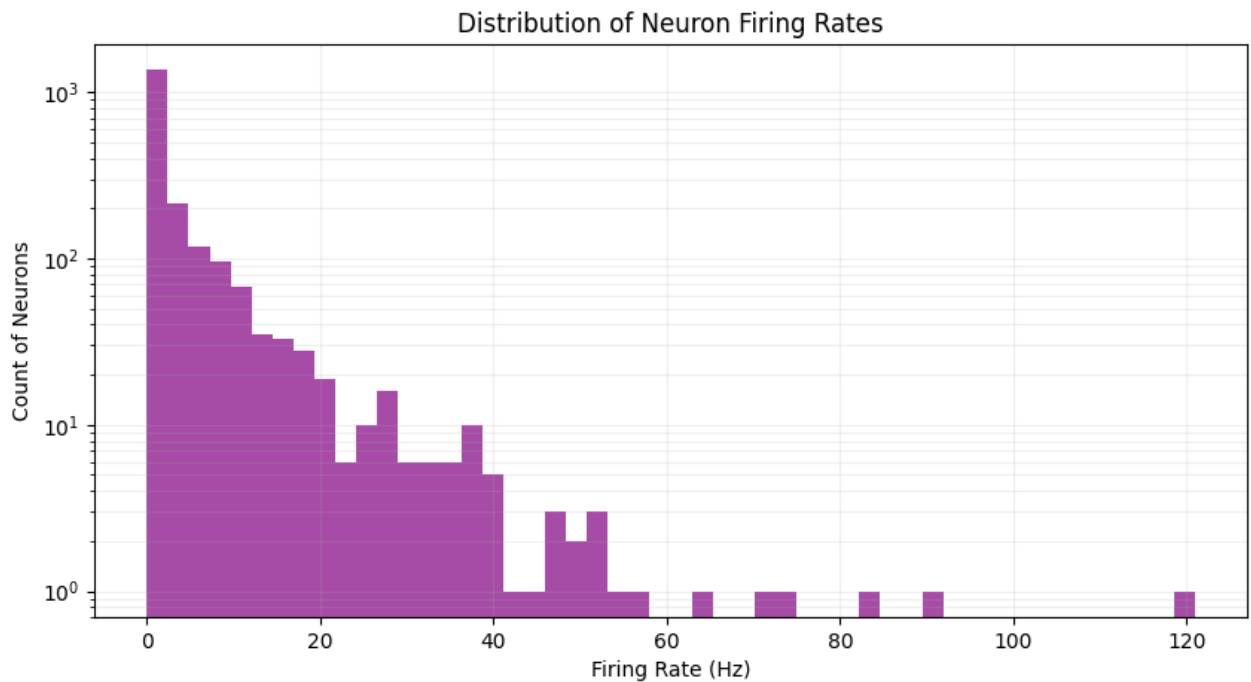
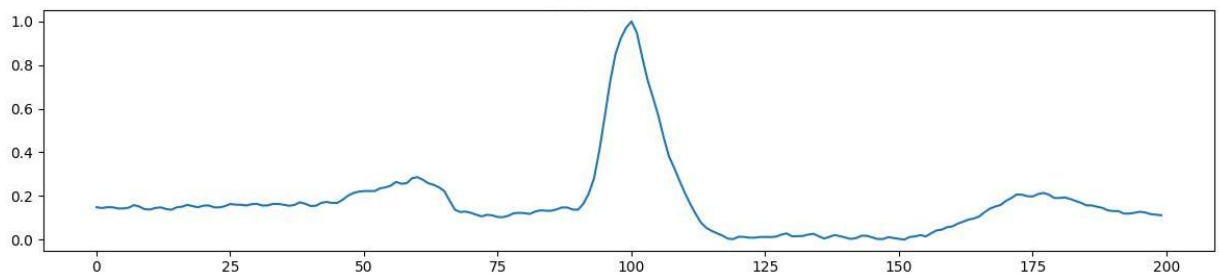


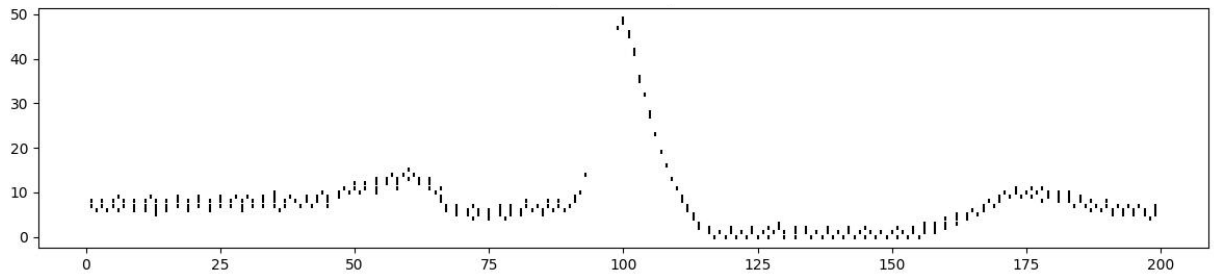
Рисунок 4.5 – Активність нейронів резервуару за інференс

Спостерігається чітке виділення QRS-комплексу вхідними спайками та розріджена, але інформативна активність резервуару, що призводить до правильної класифікації.

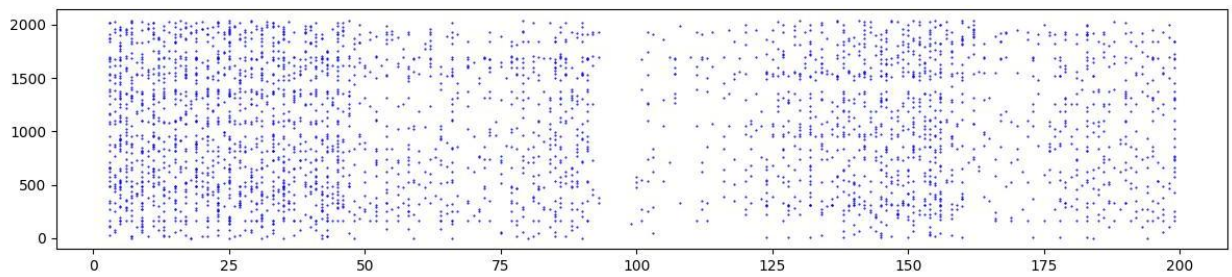
Розроблена модель продемонструвала виняткову ефективність: середня чутливість 0.9678, коефіцієнт узгодженості Каппа-Коена 0.9649, абсолютну чутливість на трьох класах аритмій та високу роздільну здатність резервуару. Результати перевищують або знаходяться на рівні найкращих публікацій для резервуарних SNN на MIT-BIH, при цьому модель зберігає ключові переваги спайкінгового кодування: розрідженість активності, низьку середню частоту спайків та високу інтерпретованість.



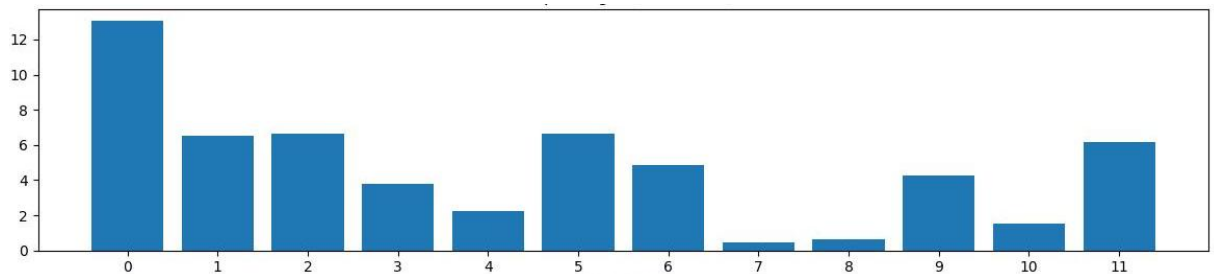
а)



б)



в)



г)

Рисунок 4.6 – Результати роботи спайкінгової нейронної мережі: а) оригінальний аналоговий сигнал ЕКГ; б) графік спайків вхідного шару; в) графік спайків «резервуару»; г) вихідні значення

Окремо був проведений експеримент на неаугментованих тестових даних для порівняння з найкращою системою класифікації ЕКГ, а саме SparrowSNN [61]. В результаті експерименту було отримано таку матрицю помилок (рис. 4.7).

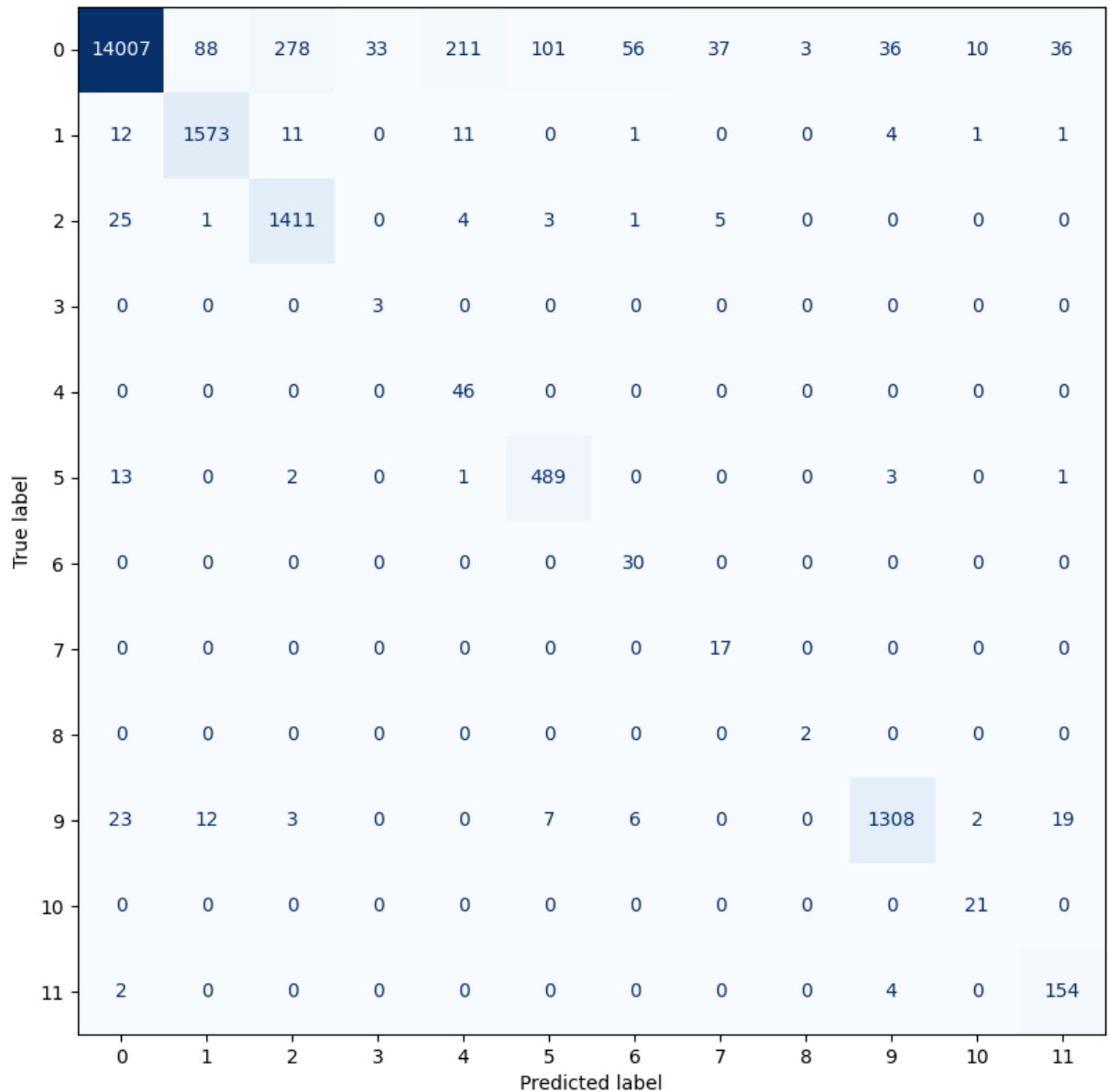


Рисунок 4.7 – Матриця помилок експерименту

Для порівняння буде продемонстрована матриця помилок для 4 класової класифікації, а саме для N, SVEB, VEB, F по AAMI. Оскільки у SparrowSNN [61] для порівняння є тільки 4-класова матриця помилок (рис. 4.8) було переформатовано матрицю з рис. 4.7 з 12 класів у 4 класи за AAMI, як у SparrowSNN [61] шляхом сумування відповідних даних:

- Клас N: N, L, R, e, j (0-4 класи)
- Клас SVEB: A, a, J, S (5-8 класи)
- Клас VEB: V, E (9-10 клас)

- Клас F: F (11 клас)

За допомогою таблиці відповідності було переведено матрицю з 12 класової класифікації у 4-класову і зображено її на рис. 4.9.

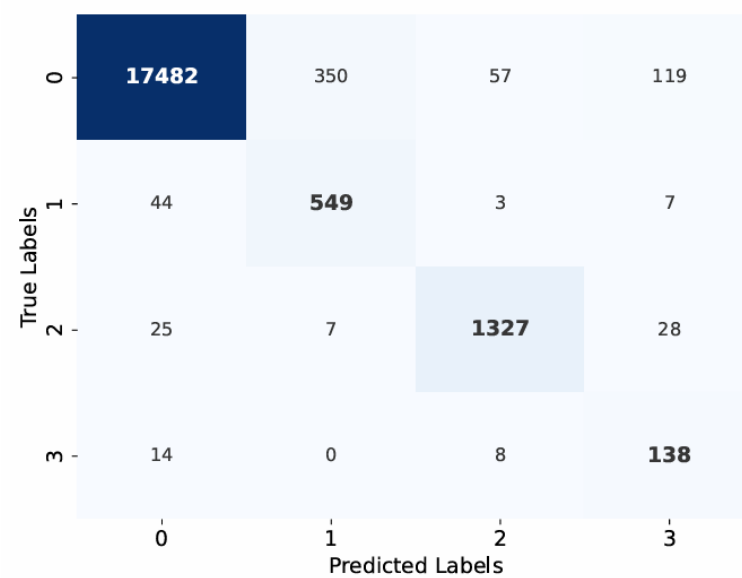


Рисунок 4.8 – Матриця помилок SparrowSNN [61]

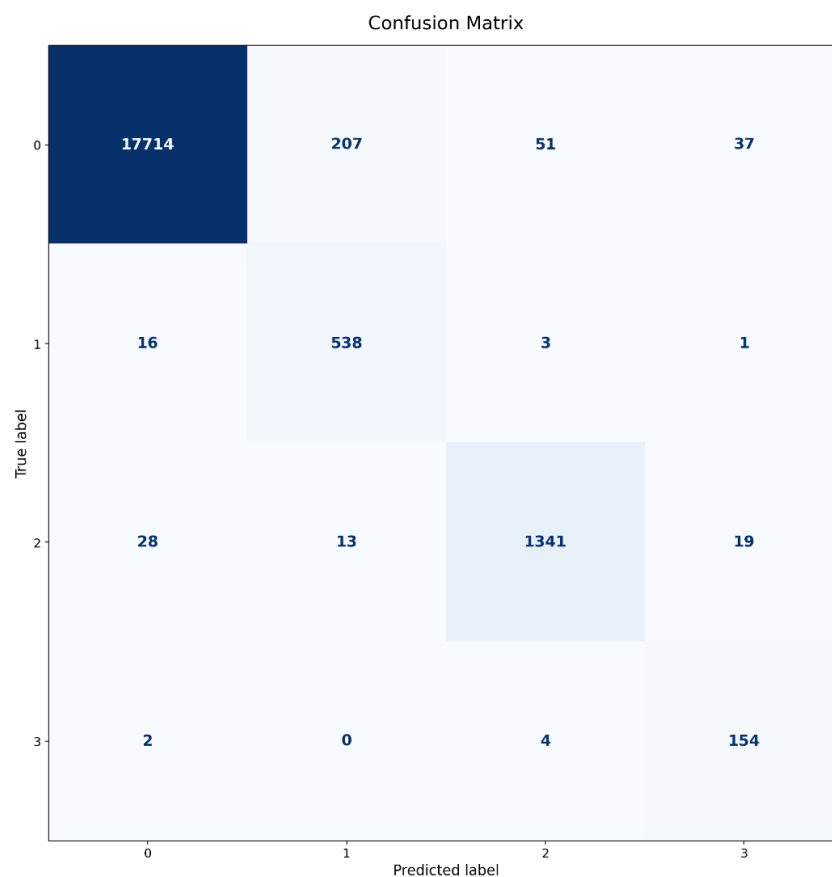


Рисунок 4.9 – Матриця помилок запропонованої системи класифікації сигналів ЕКГ

Для подальшого порівняння результатів було використано формулу чутливості та влучності для обчислення:

$$Sensitivity(Se) = \frac{TP}{TP + FN} \quad (4.7)$$

$$Precision(P+) = \frac{TP}{TP + FP} \quad (4.8)$$

та, як у [61], обчислено чутливість (в таблиці – Se) та влучність (в таблиці – P+) для кожного з 4 класів. Порівняння наведено у таблиці 4.2

Таблиця 4.2 – Порівняння роботи інформаційних технологій класифікації сигналів ЕКГ

Класифікатор	N		SVEB		VEB		F		Accuracy
	Se (%)	P+ (%)	Se (%)	P+ (%)	Se (%)	P+ (%)	Se (%)	P+ (%)	
Запропонований	98.36	99.74	96.42	70.98	95.72	95.85	96.25	72.99	98.11
SparrowSNN [61]	97.08	99.53	91.04	60.6	95.67	95.13	86.25	47.26	96.72

З табл. 4.2 видно, що запропонована інформаційна технологія класифікації сигналів ЕКГ на основі спайкінгових нейронних мереж перевершує у всіх досліджених метриках для кожного класу систему-конкурент SparrowSNN [61]. Вдалося підвищити загальну точність до 98.11%, що на 1.39% краще за аналог, та залежно від класу підвищити чутливість на 0.05-10% та влучність на 0.21-25.73%. Таким чином, у результаті проведених досліджень мета дисертаційної роботи була повністю досягнута. Запропонована інформаційна технологія на основі спайкінгових нейронних мереж забезпечує високу точність класифікації сигналів ЕКГ, перевершуючи за всіма ключовими метриками існуючі аналоги, зокрема систему SparrowSNN

[61]. Отримані результати підтверджують ефективність розробленого методу та його перспективність для практичного застосування в системах медичної діагностики.

4.4 Оцінювання ресурсоефективності та реально-часової швидкодії розробленої інформаційної технології

З урахуванням обмежень традиційних штучних нейронних мереж, які вимагають значних обчислювальних ресурсів, нова технологія повинна забезпечувати ультранизьке енергоспоживання, сумісне з носимими пристроями (наприклад, смарт-годинниками чи імплантованими датчиками), де енергія є важливим фактором для автономної роботи без підзарядки.

Для оцінки енергоспоживання розробленої системи в навантажених умовах було обрано середньодобову частоту пульсу 100 ударів за хвилину, що дозволяє перевірити енергетичні характеристики системи при підвищеному навантаженні, яке перевищує середньостатистичні показники, та тривалість безперервної роботи 24 години. При такому пульсі виникає обмеження для часу обробки інференсу у 600 мс. У якості енергоносія буде розглянуто стандартний елемент живлення CR2032 [122], який має наступні характеристики згідно специфікації Panasonic:

- Номінальна напруга: 3,0 В
- Номінальна ємність: 225 мАг (при розряді до 2,0 В)
- Номінальна енергія: Приблизно 670 мВт·год
- Стандартний струм розряду: 0,2 мА
- Максимальний імпульсний струм: До 10-15 мА (протягом коротких періодів)

Для моделювання енергоспоживання класифікації було обрано чіп Intel Loihi 2 [42]. Енергоспоживання спайкінгової нейронної мережі обчислюється за формулою [123]:

$$E_{total} = E_{SOP} * N_{SOP} + E_{spike} * N_{spike} \quad (4.8)$$

де E_{total} – це загальне енергоспоживання, E_{spike} – енергоспоживання на спайк нейрону (близько 10 пДж [123]), E_{SOP} – енергоспоживання на синаптичну операцію (близько 1 пДж [123]) – оновлення станів нейронів, що мають синапси до нейрону, що видав спайк, N_{spike} – загальна кількість спайків, N_{SOP} – загальна кількість синаптичних операцій.

У запропонованій нейронній мережі в середньому за інференс було 3768 спайків та 815081 синаптичних операцій. Загальне енергоспоживання на інференс становить близько 852,7 нДж, а потужність 1,4 мкВт. При роботі у максимальному режимі (100 ударів на хвилину) за 24 години буде спожито 5,04 мВт. При енергії батареї 670 мВт*год споживання нейронної мережі становить 0,75% від заряду батареї. Більшість холтерів [124], [125], [126], [127], [128], [129], [130], [131] працює на елементах живлення типу АА чи ААА [132], що мають більшу ємність ніж CR2032, що доводить енергоефективність розробленої системи. Порівняння енергоспоживання наведено у таб. 4.3.

Таблиця 4.3 – Порівняння енергоспоживання

Робота	Енергія на інференс	Потужність
Haoming Chu та ін. [60]	0,75 мкДж	0,93 мкВт
Alireza Amirshahi та Matin Hashemi [25]	1,78 мкДж	-
Chia Yee Saw та ін. [12]	-	0,176 Вт
Zhanglu Yan, Zhenyu Bai, Tulika Mitra та Weng-Fai Wong [61]	31 нДж	6,1 мкВт
Ця робота	0,85 мкДж	1,4 мкВт

Середній час обробки інференсу нейронною мережею становить 15-27мс. При розрахунку, що вся попередня обробка займає до 7мс, сумарний час обробки одного серцебиття становить до 34мс, що менше 200мс – мінімального інтервалу між серцебиттями при частоті пульсу 300 уд/хв, яка була обрана як максимальна на основі аналізу технічних характеристик сучасних холтерівських моніторів [124], [125], [126], [127], [128], [129], [130], [131], що доводить здатність розробленої системи до реальночасової роботи [133] та відповідно до п. 1.39 ДСТУ 2226-93 [134].

З цього можна зробити висновок, що розроблена система має високий рівень енергоефективності та достатній рівень швидкодії для роботи у реальному часі, що відповідає вимогам у розробці ресурсоефективної та реальночасової системи для класифікації сигналів ЕКГ.

4.5 Перспективи подальших досліджень

Отримані результати свідчать про високу ефективність резервуарних спайкінгових нейронних мереж для задачі класифікації ЕКГ-сигналів. Водночас розроблена технологія відкриває низку перспективних напрямків для подальшого вдосконалення та практичного впровадження.

Перенесення моделі на нейроморфне апаратне забезпечення
Наступним логічним кроком є перенесення розробленої моделі на спеціалізовані нейроморфні чіпи, такі як Intel Loihi 2 [42], IBM NorthPole [135], що дозволить реалізувати справжню подіє-керовану обробку, при якій енергія витрачається лише в момент генерації спайка. З урахуванням поточної розрідженості активності (0,84%) та середньої частоти 4,22 Гц очікується низьке енергоспоживання. Для цього планується використання фреймворків Lava (Intel) [136] або nx SDK (IBM) [137] з подальшим тестуванням на реальних чіпах.

Оптимізація моделі для носимих пристроїв Розробка версії моделі з меншим розміром резервуару (512-1024 нейронів) та застосуванням технік

квантування ваг (наприклад, до 8-бітної точності) та спайкінгового кодування (частотного або часового). Це дозволить реалізувати систему на носимих пристроях (smartwatch, Holter-монітори з Bluetooth), що працюють від акумулятора. Очікується збереження чутливості на рівні 0.94-0.96 при значному зменшенні обчислювальної складності.

Онлайн-навчання ваг вихідного шару на потокових даних Розробка адаптивного механізму оновлення тільки ваг вихідного шару у реальному часі на основі нових кардіоциклів пацієнта. Це дозволить персоналізувати модель під індивідуальні особливості ЕКГ конкретної людини (наприклад, хронічні зміни морфології QRS). Планується інтеграція з апаратними засобами безперервного моніторингу (AliveCor [138], Apple Watch ECG [139]) через Bluetooth Low Energy.

Порівняльний аналіз з найкращими моделями Проведення всебічного порівняння з сучасними ANN-архітектурами та іншими SNN-реалізаціями на одній і тій самій базі MIT-BIH. Оцінювання буде проводитися за трьома критеріями: чутливість, енергоспоживання та час інференсу. Це дозволить чітко аргументувати переваги обраного підходу.

Розширення на інші бази даних та задачі Тестування моделі на великих сучасних наборах даних: PhysioNet/CinC Challenge 2021 [140] (до 100 тис. записів), PTB-XL [141] (12-відведення). Також планується адаптація технології для виявлення інших патологій: фібриляція передсердь, ішемія, блокада пучка Гіса.

Інтеграція з реальним медичним обладнанням Розробка прототипу системи безперервного моніторингу з виведенням сповіщень на смартфон при виявленні критичних аритмій (V, S, F). Планується співпраця з виробниками медичного обладнання для сертифікації за стандартами ISO 13485 [142] та CE/FDA [143].

Дослідження біологічно більш правдоподібних моделей нейронів Перехід від простих LIF-нейронів до більш складних моделей (Іжикевич,

AdEx), що краще відтворюють біологічну динаміку. Це дозволить вивчити вплив біологічної правдоподібності на точність та енергоефективність.

Реалізація зазначених напрямків дозволить перетворити розроблену технологію з дослідницької моделі на практичний інструмент для IoT медицини та носимих пристроїв.

4.6 Висновки до четвертого розділу

Експериментальне тестування інформаційної технології класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі підтвердило її високу ефективність та практичну цінність. Описано параметри тестування та параметри середовища тестування інформаційної технології. Було розглянуто методику оцінювання точності методів класифікації сигналу ЕКГ, розроблено протокол проведення експериментальних досліджень. Було проведено експериментальні дослідження точності методу класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.

Розроблена модель досягла таких ключових результатів на тестовій вибірці з 6000 кардіоциклів: загальна точність (Accuracy) – 96,78 %; середня чутливість по класах (Macro-Recall) – 0,9678; коефіцієнт Каппа Коена – 0,9649, що відповідає «винятковому» рівню узгодженості. Детальний аналіз метрик по класах показав абсолютну чутливість ($\text{recall} = 1,00$) для трьох типів аритмій (atrial escape beat, supraventricular premature, ventricular escape beat), а також високу якість розпізнавання шлуночкових та надшлуночкових екстрасистол. Найскладнішими виявилися нормальні удари (клас 0) та передчасні надшлуночкові (клас 5), що пояснюється їхньою морфологічною схожістю з іншими класами. Візуальний аналіз (матриця помилок, ROC- та PR-криві) підтвердив мінімальну кількість помилок та високу роздільну здатність моделі. t-SNE-проекція стану резервуару продемонструвала чітке кластерне розділення всіх 12 класів, що доводить ефективність резервуарного підходу для перетворення часових сигналів на просторово роздільні патерни. Особливо важливими є нейроморфні характеристики: середня частота розрядів нейронів – 4,22 Гц; розрідженість активності – 0,84%.

Проведено додаткове дослідження для порівняння з системою аналогом. Розроблена система класифікації сигналів ЕКГ перевершила систему-конкурент SparrowSNN [61] у загальній точності на 1.39% та залежно від класу дозволила підвищити чутливість на 0.05-10% та влучність на 0.21-25.73%.

Визначено перспективи подальших досліджень, які полягають як у перенесенні моделі на нейроморфне апаратне забезпечення, оптимізації моделі для носимих пристроїв, онлайн-навчанні ваг вихідного шару на потокових даних, порівняльному аналізі з найкращими моделями, розширенням на інші бази даних та задачі, інтеграції з реальним медичним обладнанням, дослідженні біологічно більш правдоподібних моделей нейронів.

Основні наукові результати, викладені в розділі, опубліковано в роботах [5], [9], [10].

ЗАГАЛЬНІ ВИСНОВКИ

У результаті проведених досліджень було розроблено архітектуру спайкінгової нейронної мережі, яка відрізняється від відомих структурою шарів та параметрами з'єднання нейронів в резервуарі та резервуару із вхідним та вихідним шарами; удосконалено метод навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ, особливістю якого є додавання етапів кластерної ініціалізації ваг вихідних нейронів та керованої спайками корекції збуджувальних ваг цільового вихідного нейрона та гальмівних ваг інших нейронів; розроблено метод попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу, який відрізняється наявністю часової нормалізації сегментів та кодування нормалізованого сигналу в Гаусові рецептивні поля; розроблено метод класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі, який має ряд переваг перед відомими методами, зокрема, є стійким до наявних в ЕКГ шумів (викликаних, зокрема, фізіо-електричними особливостями пацієнта та нестабільністю апаратури зняття ЕКГ), не вимагає попереднього перетворення у вектор статичних ознак, має спрощений процес обробки через відсутність складних обчислювальних процедур, має покращену якість навчання спайкінгової нейронної мережі. На основі цих методів було запропоновано інформаційну технологію класифікації сигналів ЕКГ з використанням спайкінгової нейронної мережі, що в результаті призводить до підвищення точності класифікації сигналів ЕКГ при збереженні ресурсоефективності та реально-часової швидкодії.

До важливих наукових результатів, отриманих у роботі, можна віднести:

1. Вперше запропоновано метод навчання спайкінгової нейронної мережі для класифікації сигналів ЕКГ, який перед основним етапом навчання з учителем містить етап кластерної ініціалізації ваг вихідних нейронів, а після основного етапу – етап тонкого налаштування ваг нейронів вихідного шару, що забезпечило покращення якості навчання спайкінгової нейронної мережі та підвищення точності класифікації сигналів ЕКГ.

2. Удосконалено метод попередньої обробки та кодування сигналів ЕКГ для подачі у спайкінгову нейронну мережу, який відрізняється від існуючих введенням етапу часової нормалізації відліків сигналу ЕКГ у поєднанні з кодуванням у спайки на основі Гаусових рецептивних полів, що дозволило зберегти більше корисної інформації при перетворенні відліків ЕКГ у спайки і тим самим підвищити точність класифікації сигналів ЕКГ, а також дозволило зменшити середню кількість спайків на інференс і тим самим сприяти економії енергії споживання.

3. Удосконалено архітектуру спайкінгової нейронної мережі для використання у задачі класифікації сигналів ЕКГ, що відрізняється від існуючих формуванням структури нейронної мережі, вибором параметрів з'єднання нейронів між шарами та у резервуарі, що забезпечило підвищення точності класифікації сигналів ЕКГ.

4. Удосконалено інформаційну технологію класифікації сигналів ЕКГ, що відрізняється від існуючих використанням штучної спайкінгової нейронної мережі удосконаленої архітектури та спеціально розроблених методів навчання спайкінгової нейронної мережі, попередньої обробки та кодування сигналів ЕКГ, що забезпечило підвищення точності класифікації сигналів ЕКГ при збереженні ресурсоефективності та реально-часової швидкодії.

Практична цінність викладених у дисертації досліджень полягає в тому, що:

- 1) розроблено структурну схему спайкінгової нейронної мережі для використання у задачі класифікації сигналів ЕКГ;
- 2) розроблено алгоритм роботи та діаграму класів програми класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі;
- 3) розроблено програмні засоби класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі.
- 4) отримано результати тестування програми класифікації сигналів ЕКГ;

5) визначено числові значення показників точності класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі та порівняти їх із кращими відомими засобами класифікації сигналів ЕКГ.

Сукупність викладених у дисертації результатів дозволяє розв'язати важливу наукову задачу покращення точності класифікації сигналів ЕКГ засобами спайкінгових нейронних мереж, які при цьому є ресурсоефективними (споживають мінімум енергії і потребують мінімум обчислювальних ресурсів та пам'яті) та функціонують в реальному часі, тобто з мінімальною затримкою. Загалом, у відносних показниках, запропонована система за точністю на 1.39%, за чутливістю залежно від класу на 0.05 – 10% та влучністю на 0.21-25.73% краща за референсну при тестуванні на МІТ-ВІН датасеті.

Результати роботи можна використовувати в подальших дослідженнях для побудови високоточних систем класифікації сигналів ЕКГ, які характеризуються ресурсоефективністю та реальночасовою швидкодією та орієнтовані на використання у носимих пристроях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Милосердов Д. А., Колесницький О. К., Волосович О. В., Жумагулова Ш., Короленко О. В. Класифікація електрокардіограм як динамічного процесу на основі спайкінгової нейронної мережі. Оптико-електронні інформаційно-енергетичні технології. 2024. Вип. 48, № 2. С. 68–77. DOI: 10.31649/1681-7893-2024-48-2-68-77.
- [2] Милосердов Д. А., Колесницький О. К. Застосування спайкінгових нейронних мереж до класифікації електрокардіограм. Огляд. Наука і техніка сьогодні. Серія «Фізико-математичні науки». 2025. № 9(50). С. 1612–1643. DOI: 10.52058/2786-6025-2025-9(50)-1612-1643.
- [3] Милосердов Д. А., Колесницький О. К. Метод попередньої обробки сигналу ЕКГ для класифікації електрокардіограм за допомогою спайкінгових нейронних мереж. Технічні науки та технології. 2025. № 3(41). С. 203–210. DOI: 10.25140/2411-5363-2025-3(41)-203-210. URL: <http://tst.stu.cn.ua/article/view/345202/332545>.
- [4] Милосердов Д. А., Колесницький О. К. Метод класифікації електрокардіограм із застосуванням спайкінгових нейронних мереж. Вісник Вінницького політехнічного інституту. 2025. № 5. С. 121–128. DOI: 10.31649/1997-9266-2025-182-5-121-128
- [5] Милосердов Д. А., Колесницький О. К. Інформаційна технологія класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі. Наука і техніка сьогодні. Серія «Фізико-математичні науки». 2026. № 2(56). С. 2535–2552. DOI: 10.52058/2786-6025-2026-2(56)-2535-2552.
- [6] Семчук В. О., Милосердов Д. А., Колесницький О. К. Інформаційна технологія розпізнавання вимовлених літер на основі спайкінгової нейронної мережі. Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)». Вінниця, 2023. [Електронний ресурс]. URL:

<https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/16854/1405>

5.

[7] Милосердов Д. А., Колесницький О. К. Розпізнавання серцевої недостатності як багатопараметричного процесу на основі спайкінгових нейронних мереж. Матеріали 1-ої Міжнародної науково-практичної конференції «Інформаційні системи та технології: результати і перспективи (IST 2024)», 6 березня 2024 р. Київ : ФІТ КНУТШ, 2024. [Електронний ресурс]. URL:

https://ist.fit.knu.ua/_files/ugd/016074_8fc56ecbbfa94cde874eedd872f3780e.pdf.

[8] Милосердов Д. А., Колесницький О. К. Метод попередньої обробки сигналу електрокардіограм для класифікації за допомогою спайкінгових нейронних мереж. Science, Technology and Industry in the Digital Age : Collection of Scientific Papers with Proceedings of the 1st International Scientific and Practical Conference. International Scientific Unity. September 24–26, 2025. Hamburg, Germany. P. 159–164. DOI: 10.70286/isu-24.09.2025.

[9] Милосердов Д. А., Колесницький О. К. Інформаційна технологія класифікації сигналів ЕКГ за допомогою спайкінгових нейронних мереж. Innovative Research in Science and Economy : Collection of Scientific Papers with Proceedings of the 4th International Scientific and Practical Conference. International Scientific Unity. Brussels, Belgium. April 1–3, 2026. P. 178–181. DOI: 10.70286/ISU-01.04.2026.009.

[10] Свідоцтво про реєстрацію авторського права на твір № 139885. Комп'ютерна програма «Інформаційна технологія класифікації часових рядів». Колесницький О. К., Денисов І. К., Милосердов Д. А. Дата реєстрації: 06.10.2025.

[11] World Heart Federation. World Heart Report 2023: Confronting the World's Number One Killer. Geneva, Switzerland, 2023. [Електронний ресурс]. URL: <https://world-heart-federation.org/wp-content/uploads/World-Heart-Report-2023.pdf>.

[12] Saw C. Y., Wong Y. C. Neuromorphic computing based on stochastic

spiking reservoir for heartbeat classification. *Jordanian J. Comput. Inf. Technol.* 2022. Vol. 8, no. 2. P. 184–195. DOI: 10.5455/jjcit.71-1642596471.

[13] Rana A., Kim K. K. Electrocardiography classification with leaky integrate-and-fire neurons in an artificial neural network-inspired spiking neural network framework. *Sensors*. 2024. Vol. 24, no. 10. P. 3246. DOI: 10.3390/s24103246.

[14] Mohammad D. Detection of abnormalities in cardiac rhythm using spiking neural networks : M.S. thesis / Dept. Elect. Inf. Technol. Lund Univ. Lund, Sweden, 2023.

[15] Карась О. В., Заболотна Н. І., Павлов С. В. Аналіз роботи системи підтримки прийняття рішень на основі нейромережі для медичного діагностування. *Оптоелектронні інформаційно-енергетичні технології*. 2020. № 1. С. 38–44.

[16] Мартинюк Т. Б., Запетрук Я. В. Нейромережевий підхід до медичної експрес-діагностики. *Вісник Вінницького політехнічного інституту*. 2019. Вип. 6. С. 37–44.

[17] Chaikovsky I, Senko I, Budnyk M, Matsyshyn V, Ryzhenko T, Budnyk V, Romanchuk O, Popov A, Stetsyuk P. Analysis of heart rate variability and subtle ECG changes based on machine learning for objective assessment of the psychological state of military personnel. *Front Psychol.* 2026 Feb 27;17:1688230. doi: 10.3389/fpsyg.2026.1688230. PMID: 41835868; PMCID: PMC12982173.

[18] Chaikovsky I, Kohler J, Hecker T, Hailer B, Auth-Eisernitz S, Sosnytsky V, Feinzelberg L, Budnik N and Steinberg F, Detection of coronary artery disease in patients with normal or unspecifically changed ECG on the basis of magnetocardiography *Biomag 2000: Proc. 12th Int. Conf. on Biomagnetism (Espoo, Finland, 13–17 August 2000)* ed J Nenonen et al. (Helsinki University of Technology) p 565

[19] Krak, I., Pashko A., Barmak O., Pavlov S. Selection parameters in the ECG signals for analysis of QRS complexes, *Ceur Workshop Proceedings*, 2020, 2623, pp. 1–13.

[20] О. В. Ковальчук і О. Бармак, «Метод класифікації аритмій на ЕКГ-сигналі», *Опт-ел. інф-енерг. техн.*, вип. 48, вип. 2, с. 34–44, Лис 2024.

[21] Інтелектуальні технології в медичній діагностиці, лікуванні та реабілітації: монографія / [С. В. Павлова, О.Г. Авруніна, С.М.Злепка, Є.В.Бодянського, І.А. Чайковського та ін.]; за редакцією С.Павлова, О.Авруніна. – Вінниця: ПП «ТД «Едельвейс і К», 2019. –260 с. ISBN 978-617-7237-59-3

[22] Злепка С. М. Апаратура для фізіотерапії та діагностики: навч. посіб. / [С. М. Злепка, С. В. Павлов, В. Б. Василенко, та ін.]. – Вінниця : ВНТУ, 2011. – 212 с.

[23] Choi H., Park J., Lee J., Sim D. Review on spiking neural network-based ECG classification methods for low-power environments. *Biomed. Eng. Lett.* 2024. Vol. 14, no. 4. P. 829–841. DOI: 10.1007/s13534-024-00391-2.

[24] Yan Z., Zhou J., Wong W.-F. Energy efficient ECG classification with spiking neural network. *Biomed. Signal Process. Control.* 2021. Vol. 63. P. 102170. DOI: 10.1016/j.bspc.2020.102170.

[25] Amirshahi A., Hashemi M. ECG classification algorithm based on STDP and R-STDP neural networks for real-time monitoring on ultra low-power personal wearable devices. *IEEE Trans. Biomed. Circuits Syst.* 2019. Vol. 13, no. 6. P. 1483–1493. DOI: 10.1109/TBCAS.2019.2948920.

[26] Al-khrai Jah M., Alruwaili M., Alruwaili A., Alhejaili A., Alrashedi A., Alruwaili A. Cardiovascular disease and covid-19: A systematic review. *Int. J. Cardiol. Cardiovasc. Risk Prev.* 2024. Vol. 22. P. 200301. DOI: 10.1016/j.ijcrp.2024.200301.

[27] Yan Z., Bai Z., Mitra T., Wong W.-F. SparrowSNN: A hardware/software co-design for energy efficient ECG classification. *arXiv:2406.06543 [cs.AR]*. 2024.

[28] Statista Research Department. Wearables - statistics & facts. Statista. Hamburg, Germany, Nov. 2023. [Електронний ресурс]. URL: <https://www.statista.com/topics/1556/wearable-technology/>.

- [29] Banerjee D., Dey S., Pal A. An SNN based ECG classifier for wearable edge devices. Proc. Workshop Learning Time Series Health, 36th Conf. Neural Inf. Process. Syst. (NeurIPS). New Orleans, LA, USA, 2022. P. 1–7.
- [30] Kovács P., Samiee K. Arrhythmia detection using spiking variable projection neural networks. Proc. Computing Cardiology (CinC). Vol. 49. Tampere, Finland, Sep. 2022. P. 1–4. DOI: 10.22489/CinC.2022.049.
- [31] Boashash B. (ed.). Time–Frequency Signal Analysis and Processing: A Comprehensive Reference. Oxford : Elsevier Science, 2003. ISBN 978-0-08-044335-5.
- [32] Arif M. Robust electrocardiogram (ECG) beat classification using discrete wavelet transform. Physiol. Meas. 2008. Vol. 29, no. 5. P. 555.
- [33] Wold S., Esbensen K., Geladi P. Principal component analysis. Chemom. Intell. Lab. Syst. 1987. Vol. 2, no. 1–3. P. 37–52.
- [34] Comon P. Independent component analysis, a new concept?. Signal Process. 1994. Vol. 36, no. 3. P. 287–314.
- [35] Т. Б. Мартинюк, А. В. Кожем'яко, Д. О. Каташинський, і І. В. Булига, «ОСОБЛИВОСТІ ПРОЦЕСУ КЛАСИФІКАЦІЇ У КОНТЕКСТІ МЕДИЧНОГО ДІАГНОСТУВАННЯ», НаукПраці ВНТУ, вип. 1, Бер 2025.
- [36] Van The Huy, Duong Tuan Anh. An efficient implementation of anytime k-medoids clustering for time series under dynamic time warping. Proceedings of the Seventh Symposium on Information and Communication Technology, SoICT 2016. Ho Chi Minh City, Vietnam, December 8–9, 2016.
- [37] Bardachenko V. F., Kolesnytskyi O. K., Vasiletskyi S. A. Prospects for the application of impulse neural networks with a timer representation of information for the recognition of dynamic images. UsiM. 2003. No. 6. P. 73–82.
- [38] Kolesnytskyj O. K., Bokotsey I. V., Yaremchuk S. S. Optoelectronic implementation of pulsed neurons and neural networks using bispin-devices. Optical Memory & Neural Networks (Information Optics). 2010. Vol. 19, no. 2. P. 154–165.
- [39] Ji S., Xu W., Yang M., Yu K. 3D convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 2013. Vol. 35,

no. 1. P. 221–231. ISSN 0162-8828.

[40] Guo L., Sim G., Matuszewski B. Inter-patient ECG classification with convolutional and recurrent neural networks. *Biocybern. Biomed. Eng.* 2019. Vol. 39, no. 3. P. 868–879.

[41] Moody G. B., Mark R. G. The impact of the MIT-BIH arrhythmia database. *IEEE Eng. Med. Biol. Mag.* 2001. Vol. 20, no. 3. P. 45–50.

[42] Lin C.-K., Wild A., Chinya G. N., Cao Y., Davies M., Lavery D. M., Wang H. Programming spiking neural networks on Intel's Loihi. *Computer*. 2018. Vol. 51, no. 3. P. 52–61.

[43] Xing Y., Zhang G., Shang J., Liu L., Sun G. Accurate ECG classification based on spiking neural network and attentional mechanism for real-time implementation on personal portable devices. *Electronics*. 2022. Vol. 11, no. 12. P. 1889.

[44] Arora A., Basha S. H. S., McCormick M. E., Rahimi A., Farkhani S. H. A hybrid spiking neural network–transformer architecture for motor imagery and sleep apnea detection. *Frontiers Neurosci.* 2025. Vol. 19.

[45] Bai Y., Zhou Y., Li R. A novel spiking neural network for ECG signal classification. *Proc. Int. Conf. Comput. Intell. Neurosci.* 2025. P. 1–6.

[46] Ghosh-Dastidar S., Adeli H. The promise of spiking neural networks for ubiquitous computing: A survey and new perspectives. *arXiv:2506.01737*. 2025.

[47] Kim J., Park S., Lee H. QCSNN: A memory-efficient spiking neural network for on-device ECG-based arrhythmia detection. *Proc. IEEE Int. Conf. Comput. Intell. Netw. Syst.* 2025. P. 1–8.

[48] Amirshahi A., Hashemi M. Exploring the potential of spiking neural networks in biomedical applications: Advantages, limitations, and future perspectives. *Front. Neurosci.* 2024. Vol. 18.

[49] Яровий А. А. Методи та засоби організації високопродуктивних паралельно-ієрархічних обчислювальних систем із рекурсивною архітектурою: монографія. Вінниця: ВНТУ, 2016.

[50] Шемет, Є. О., Т. О. Шемет, and А. А. Яровий. Проблемні аспекти

класифікації зображень на основі згорткових нейронних мереж. Diss. ВНТУ, 2019.

[51] Rangappa V. G., Prasad S., Agarwal A. Classification of cardiac arrhythmia stages using hybrid features extraction with K-nearest neighbour classifier of ECG signals. *Learning*. 2018. Vol. 11. P. 21–32.

[52] Savalia S., Emamian V. Cardiac arrhythmia classification by multi-layer perceptron and convolution neural networks. *Bioengineering*. 2018. Vol. 5, no. 2. P. 35.

[53] Mastoi Q., Wah T. Y., Raj R. G., Lakhan A. A novel cost-efficient framework for critical heartbeat task scheduling using the Internet of Medical Things in a fog cloud system. *Sensors*. 2020. Vol. 20, no. 2. P. 441.

[54] Jiang C., Fan T., Gao H., Shi W., Liu L., Cérin C., Wan J. Energy aware edge computing: A survey. *Computer Communications*. 2020. Vol. 151. P. 556–580.

[55] Brainchip. Brainchip Unveils the Akida Development Environment. 2019. [Електронний ресурс]. URL: <https://www.brainchipinc.com/newsmedia/press-releases/detail/61/brainchip-unveils-the-akida-development-environment>.

[56] World Health Organization. Cardiovascular diseases (CVDs). Geneva, Switzerland, 2019. [Електронний ресурс]. URL: [https://www.who.int/news-room/factsheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/factsheets/detail/cardiovascular-diseases-(cvds)).

[57] Jaeger H. Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "Echo State Network" approach. *GMD-Forschungszentrum Informationstechnik Bonn*. 2002. Vol. 5, no. 01.

[58] Ponulak F., Kasinski A. Introduction to spiking neural networks: information processing, learning and applications. *Acta Neurobiologiae Experimentalis*. 2011. Vol. 71, no. 4. P. 409–433.

[59] Rana A., Kim K. K. A novel spiking neural network for ECG signal classification. *Journal of Sensor Science and Technology*. 2021. Vol. 30, no. 1. P. 20–24.

[60] Chu H., Yan Y. A neuromorphic processing system with spike-driven

SNN processor for wearable ECG classification. *IEEE Trans. Biomed. Circuits Syst.* 2022. Vol. 16, no. 4.

[61] Yan Z., Bai Z., Mitra T., Wong W.-F. SparrowSNN: A hardware/software co-design for energy efficient ECG classification. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2018. Vol. 37, no. 4. Art. 111. 22 p.

[62] Liu, A., Azmine, M. F., Lin, C., & Yi, Y. (2026). Reservoir Computing: Foundations, Advances, and Challenges Toward Neuromorphic Intelligence. *AI*, 7(2), 70. <https://doi.org/10.3390/ai7020070>

[63] Lagani G., Falchi F., Gennaro C., Amato G. Spiking neural networks and bio-inspired supervised deep learning: A survey. 2018. Vol. 1. P. 31.

[64] Yin B., Corradi F., Bohté S. M. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence.* 2021. Vol. 3, no. 10. P. 905–913. DOI: 10.1038/s42256-021-00397-w.

[65] Li J., Yang S. X. A novel feature learning-based bio-inspired neural network for real-time collision-free rescue of multi-robot systems. *IEEE Trans. Ind. Electron.* 2024. DOI: 10.1109/TIE.2024.3370939.

[66] Barranca V. J. et al. Dynamics of the exponential integrate-and-fire model with slow currents and adaptation. *J. Comput. Neurosci.* 2013. Vol. 37. P. 161–180. DOI: 10.1007/s10827-013-0494-0.

[67] Xu Q., Li Y., Shen J. et al. Hierarchical spiking-based model for efficient image classification with enhanced feature extraction and encoding. *IEEE Trans. Neural Netw. Learn. Syst.* 2022. P. 1–9. DOI: 10.1109/tnnls.2022.3232106.

[68] Ren D., Ma Z., Chen Y., Peng W., Liu X., Zhang Y., Guo Y. Spiking PointNet: Spiking Neural Networks for Point Clouds. *arXiv.* 2023. DOI: 10.48550/arXiv.2310.06232.

[69] Xiao M., Meng Q., Zhang Z., He D., Lin Z. Online training through time for spiking neural networks. *arXiv.* 2022. DOI: 10.48550/arXiv.2210.04195.

[70] Qiu X., Zhu R.-J., Chou Y., Wang Z., Deng L.-J., Li G. Gated attention coding for training high-performance and efficient spiking neural networks. *arXiv.* 2024. DOI: 10.48550/arXiv.2308.06582.

- [71] Izhikevich E. M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 2003. Vol. 14, no. 6. P. 1569–1572.
- [72] İlknur Kayıkçıoğlu Bozkır, Zübeyir Özcan, Cemal Köse et al. Classification of a Two-Class ECG Dataset Based on Perceptron Learning in a Cortical Pyramidal Neuron Model, 2022, Research Square <https://doi.org/10.21203/rs.3.rs-2149891/v1>.
- [73] Dampfhoffer M., Mesquida T., Valentian A., Anghel L. Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* 2023. P. 1–16. DOI: 10.1109/TNNLS.2023.3263008.
- [74] Soures N., Kudithipudi D. Deep liquid state machines with neural plasticity for video activity recognition. *Frontiers in Neuroscience*. 2019. Vol. 13. P. 686. DOI: 10.3389/fnins.2019.00686.
- [75] Kudithipudi D., Saleh Q., Merkel C., Thesing J., Wysocki B. Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing. *Frontiers in Neuroscience*. 2016. Vol. 9. P. 502. DOI: 10.3389/fnins.2015.00502.
- [76] Suárez L. E. et al. conn2res: A toolbox for connectome-based reservoir computing. 2023. *bioRxiv* 2023.05.31.543092; doi: <https://doi.org/10.1101/2023.05.31.543092>.
- [77] Tavanaei A., Ghodrati M., Kheradpisheh S. R., Masquelier T., Maida A. Deep learning in spiking neural networks. *Neural Networks*. 2018. [Электронный ресурс]. URL: <https://arxiv.org/abs/1804.08150v4>.
- [78] Werbos P. J. Backpropagation through time: What it is and how to do it. *Proc. IEEE*. 1990. Vol. 78, no. 10. P. 1550–1560.
- [79] De Jesús O., Hagan M. Backpropagation algorithms for a broad class of dynamic networks. *IEEE Trans. Neural Netw.* 2007. Vol. 18, no. 1. P. 14–43.
- [80] Williams R. J., Zipser D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*. 1989. Vol. 1, no. 2. P. 270–280.

- [81] Irie K., Gopalakrishnan A., Schmidhuber J. Exploring the promise and limits of real-time recurrent learning. arXiv:2305.19044. 2023.
- [82] Yada Y., Yasuda S., Takahashi H. Physical reservoir computing with FORCE learning in a living neuronal culture. *Appl. Phys. Lett.* 2021. Vol. 119, no. 17. P. 173701.
- [83] Sussillo D., Abbott L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron*. 2009. Vol. 63, no. 4. P. 544–557.
- [84] Stanley K. O., Miikkulainen R. Evolving neural networks through augmenting topologies. *Evolutionary Computation*. 2002. Vol. 10, no. 2. P. 99–127.
- [85] Ackley D. H., Hinton G. E., Sejnowski T. J. A learning algorithm for Boltzmann machines. *Cognitive Science*. 1985. Vol. 9, no. 1. P. 147–169.
- [86] Ravichandran N. B., Lansner A., Herman P. Learning representations in Bayesian Confidence Propagation neural networks. *Proc. 2020 International Joint Conference on Neural Networks (IJCNN)*. 2020. P. 1–7.
- [87] Задачин В. М. Чисельні методи: навч. посіб. / В. М. Задачин. – Вінниця: ВНТУ, 2014. – 180 с.
- [88] Malcolm K., Casco-Rodriguez J. A comprehensive review of spiking neural networks: Interpretation, optimization, efficiency, and best practices. 2023. [Електронний ресурс]. URL: <https://arxiv.org/abs/2303.10780v2>.
- [89] Auge D., Hille J., Mueller E., Knoll A. A survey of encoding techniques for signal processing in spiking neural networks. *Neural Processing Letters*. 2021. Vol. 53. P. 4693–4710. DOI: 10.1007/s11063-021-10562-2.
- [90] Barchid S., Mennesson J., Eshraghian J., Djéraba C., Bennamoun M. Spiking neural networks for frame-based and event-based single object localization. 2022. [Електронний ресурс]. URL: <https://arxiv.org/abs/2206.06506>.
- [91] Pagkalos M., Chavlis S., Poirazi P. Introducing the Dendriify framework for incorporating dendrites to spiking neural networks. *Nature Communications*. 2023. Vol. 14, no. 1. P. 131. DOI: 10.1038/s41467-022-35747-8.
- [92] Maršánová L., Němcová A., Smíšek R., Vitek M., Smítal L. Advanced P wave detection in ECG signals during pathology: Evaluation in different

arrhythmia contexts. *Scientific Reports*. 2019. Vol. 9. P. 19053. DOI: 10.1038/s41598-019-55323-3.

[93] Scrugli M. A., Busia P., Leone G., Meloni P. On-FPGA spiking neural networks for integrated near-sensor ECG analysis. 2024.

[94] Khalaf A. J., Mohammed S. J. Verification and comparison of MIT-BIH arrhythmia database based on number of beats. *Int. J. Electr. Comput. Eng. (IJECE)*. 2021. Vol. 11, no. 6. P. 4950–4961. DOI: 10.11591/ijece.v11i6.pp4950-4961.

[95] Huayu L., Gregory D., Janet R., Ao L. DeScoD-ECG: Deep score-based diffusion model for ECG baseline wander and noise removal. 2023. [Электронный ресурс]. URL: <https://arxiv.org/abs/2208.00542v2>.

[96] Van Rossum G., de Boer J. Interactively testing remote servers using the Python programming language. *CWI Quarterly*. 1991. Vol. 4, no. 4. P. 283–303.

[97] Higham D. J., Higham N. J. A history of MATLAB. *Proc. ACM Program. Lang.* 2020. Vol. 4. DOI: 10.1145/3386331.

[98] Bishnu S. et al. Comparing the performance of Julia on CPUs versus GPUs. *Geoscientific Model Development*. 2023. Vol. 16. P. 5539–5559. DOI: 10.5194/gmd-16-5539-2023.

[99] Plauska I. et al. Performance evaluation of C/C++, MicroPython, Rust and other languages for embedded systems. *Electronics*. 2022. Vol. 12, no. 1. DOI: 10.3390/electronics12010143.

[100] Python Software Foundation ; PhysioNet. WFDB Python package. 2022. [Электронный ресурс]. URL: <https://pypi.org/project/wfdb/>.

[101] Bota P., Silva R., Carreiras C., Fred A., Plácido da Silva H. BioSPPy: A Python toolbox for physiological signal processing. *SoftwareX*. 2024. Vol. 26. P. 101712. DOI: 10.1016/j.softx.2024.101712.

[102] Van Gent P., Farah H., van Nes N., van Arem B. HeartPy: A novel heart rate algorithm for the analysis of noisy signals. *Transp. Res. Part F Traffic Psychol. Behav.* 2019. Vol. 66. P. 368–378.

[103] Makowski D., Pham T., Lau Z. J., Brammer J. C., Lespinasse F., Pham

H., Schölzel C., Chen S. H. A. NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behav. Res. Methods*. 2021. Vol. 53, no. 4. P. 1689–1696. DOI: 10.3758/s13428-020-01516-y.

[104] Hunter J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* 2007. Vol. 9, no. 3. P. 90–95.

[105] Waskom M. Seaborn: Statistical data visualization. *J. Open Source Softw.* 2021. Vol. 6, no. 60. P. 3021.

[106] Plotly Technologies Inc. Collaborative data science. Montréal, QC : Plotly Technologies Inc., 2015. [Электронный ресурс]. URL: <https://plot.ly>.

[107] Bokeh Development Team. Bokeh: Python library for interactive visualization. 2018. [Электронный ресурс]. URL: <http://www.bokeh.pydata.org>.

[108] Fei N., Gao Y., Lu Z., Xiang T. Z-score normalization, hubness, and few-shot learning. *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*. 2021. P. 142–151.

[109] Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* 2002. Vol. 16. P. 321–357.

[110] Turner A. (ed.). What's new in Python 3.12. Python Software Foundation, Oct. 2023. [Электронный ресурс]. URL: <https://docs.python.org/3/whatsnew/3.12.html>.

[111] Paszke A. et al. PyTorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst. (NeurIPS)*. 2019. P. 8024–8035.

[112] Fang W. et al. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Sci. Adv.* 2023. Vol. 9, no. 40. Art. no. eadi1480.

[113] NumPy developers. NumPy 1.24.0 release notes. NumPy, Jan. 2023. [Электронный ресурс]. URL: <https://numpy.org/doc/stable/release/1.24.0-notes.html>.

[114] SciPy developers. SciPy 1.10.0 release notes. SciPy, Dec. 2022. [Электронный ресурс]. URL: <https://docs.scipy.org/doc/scipy/release/1.10.0->

notes.html.

[115] Pedregosa F. et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 2011. Vol. 12. P. 2825–2830.

[116] Fawcett T. An introduction to ROC analysis. *Pattern Recognit. Lett.* 2006. Vol. 27, no. 8. P. 861–874.

[117] Sokolova M., Lapalme G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* 2009. Vol. 45, no. 4. P. 427–437.

[118] Cohen J. A coefficient of agreement for nominal scales. *Educ. Psychol. Meas.* 1960. Vol. 20, no. 1. P. 37–46.

[119] Visa S., Ramsay A., Shaikh A., Vitoroulis T. Confusion matrix visualization. *Proc. Int. Conf. Data Min. (DMIN)*. 2011. P. 1–7.

[120] Davis J., Goadrich M. The relationship between precision-recall and ROC curves. *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*. 2006. P. 233–240.

[121] van der Maaten L., Hinton G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 2008. Vol. 9. P. 2579–2605.

[122] Panasonic Industrial. AAA4000C321 Datasheet. [Электронный ресурс]. URL: <https://industrial.panasonic.com/cdbs/www-data/pdf2/AAA4000/AAA4000C321.pdf>.

[123] Laitinen Imanov, Yunus O., Kulali D., Erisken D., Turhan R. Energy-efficient neuromorphic computing for edge AI: A framework with adaptive spiking neural networks and hardware-aware optimization. *arXiv*. 2026. DOI: 10.48550/arXiv.2602.02439.

[124] Martínez-Suárez F., García-Limón J. A., Baños-Bautista J. E., Alvarado-Serrano C., Casas O. Low-power long-term ambulatory electrocardiography monitor of three leads with beat-to-beat heart rate measurement in real time. *Sensors*. 2023. Vol. 23, no. 19. P. 8303.

[125] NorthEast Monitoring, Inc. DR181 Digital Holter Recorder Operator's Manual. Maynard, MA, USA, Jan. 2016.

[126] Advanced Instrumentations. Cardiology Monitor HT-1000 Technical

Specifications. Miami, FL, USA, 2018.

[127] Fourth Frontier. Next-Gen Holter Monitoring: Frontier X Plus. 2026.

[128] Xpart. How to Use the ADS1292R ECG/Respiration Breakout Kit for Accurate Health Monitoring. Aug. 28, 2024.

[129] Cardioline SpA. Walk400h/Clickholter Operating Manual. Rev. 09. Trento, Italy, Feb. 2, 2023.

[130] ТОВ «Компанія «СІАТА». Система холтерівського моніторингу ЕКГ ECGpro Holter (версія Lite). [Електронний ресурс]. URL: https://storage.ua.prom.st/2151098_holter_ecgpro_dmat_abppro_ciata.pdf

[131] Холтер АТ ВАТ41-2 [Електронний ресурс]. ТехноМед : офіційний сайт. – URL: <https://techno-med.com.ua/ua/holter-ad-vat41-2.html>.

[132] Farnell. Datasheet. [Електронний ресурс]. URL: <https://www.farnell.com/datasheets/2626068.pdf>.

[133] В.Г. Зайцев, Є.І. Цибаєв, КОМП'ЮТЕРНІ СИСТЕМИ РЕАЛЬНОГО ЧАСУ, НАВЧАЛЬНИЙ ПОСІБНИК КПІ ім. Ігоря Сікорського, 2019.

[134] ДСТУ 2226-93. Автоматизовані системи. Терміни та визначення. – [Чинний від 1994-07-01]. – Київ : Держстандарт України, 1994. – 48 с.

[135] IBM Research. NorthPole: An energy-efficient, AI-first processor. IBM, 2023. [Електронний ресурс]. URL: <https://research.ibm.com/blog/northpole-ai-processor>.

[136] Intel Labs. Lava: Open-source software framework for neuromorphic computing. Intel, 2022. [Електронний ресурс]. URL: <https://lava-nc.org/>.

[137] IBM Research. IBM Neural Explorer (nx) SDK. 2023–2024. [Електронний ресурс]. URL: <https://research.ibm.com/projects/neural-explorer>.

[138] AliveCor. AliveCor - офіційна сторінка. [Електронний ресурс]. URL: <https://alivecor.com/uk>.

[139] Apple Inc. Apple Watch - ECG App. [Електронний ресурс]. URL: <https://support.apple.com/uk-ua/120278>.

[140] Computing in Cardiology (CinC). PhysioNet/CinC Challenge 2021:

Willing and able - translation of physiological data. 2021. [Электронный ресурс]. URL: <https://physionet.org/content/challenge-2021/>.

[141] Wagner P., Strodthoff N., Bousseljot R., Samek W., Schaeffter T. PTB-XL, a large publicly available electrocardiography dataset (version 1.0.3). PhysioNet. 2022. DOI: 10.13026/kfzx-aw45.

[142] International Organization for Standardization. ISO 13485:2016 Medical devices - Quality management systems - Requirements for regulatory purposes. Geneva, Switzerland, 2016.

[143] European Commission. CE marking. [Электронный ресурс]. URL: https://ec.europa.eu/growth/single-market/ce-marking_en.

ДОДАТКИ

Додаток А Акти впровадження результатів дисертації



АКТ

про впровадження результатів дисертаційної роботи
**«Інформаційна технологія на основі спайкінгових нейронних мереж для
 класифікації сигналів ЕКГ»**
 Милосердова Дмитра Андрійовича

Комісія у складі:

Голова комісії:

Директор ТОВ «ВІНСТАРС ТЕХНОЛОДЖІ» Софіна Дмитро Васильович

Члени комісії:

Комерційний директор ТОВ «ВІНСТАРС ТЕХНОЛОДЖІ» Гуменчук Юрій
 Олександрович;

Фахівець з розроблення комп'ютерних програм та програмного забезпечення
 Кучук Олексій Миколайович.

Комісія підтверджує, що результати наукових досліджень, які особисто отримані Милосердовим Дмитром Андрійовичем у дисертаційному дослідженні, використовуються ТОВ «ВІНСТАРС ТЕХНОЛОДЖІ» при роботі з поточними та перспективними проектами, що пов'язані з нейронними мережами, а саме:

- метод навчання спайкінгової нейронної мережі з використанням етапу кластерної ініціалізації ваг вихідних нейронів та етапу тонкого налаштування;
- удосконалений метод попередньої обробки та кодування часових сигналів у спайки;
- удосконалену архітектуру спайкінгової нейронної мережі, адаптовану для роботи з динамічними сигналами.

Впровадження наукових і практичних результатів дозволило сформувані заходи підвищення точності класифікації динамічних сигналів, зниженню енергоспоживання моделі.

Голова комісії

Дмитро СОФИНА

Члени комісії:

Юрій ГУМЕНЧУК

Олексій КУЧУК

«ЗАТВЕРДЖУЮ»
 Директор ПМП «Центр-В»

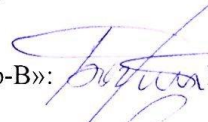
 Багрій Дмитро Анатолійович
 «16» 09 2026 р.

АКТ

впровадження результатів дисертаційної роботи
**«Інформаційна технологія на основі спайкінгових нейронних мереж для
 класифікації сигналів ЕКГ»**
 Милосердова Дмитра Андрійовича

Комісія ПМП «Центр-В» у складі: Багрія Д. А., Лихвар Н. О., Кириченко О. В. розглянула запропонований у даній науковій роботі метод класифікації сигналів ЕКГ, у якому завдяки використанню спайкінгової нейронної мережі на основі резервуарного числення та методу попередньої обробки та кодування сигналу ЕКГ збережена ресурсоефективність для роботи на носимому офлайн пристрої холтерівського моніторингу та має високу точність класифікації сигналів ЕКГ. Це дозволить підвищити інтеграцію систем моніторингу у медичній IoT та допоможе збирати більше інформації для клінічних досліджень.

На основі вищезазначеного комісія вважає за доцільне використання результатів досліджень, отриманих у роботі Милосердова Д. А. у діяльності нашої компанії при розробці систем реального часу моніторингу ЕКГ людини.

Директор ПМП «Центр-В»:  Багрій Д. А.

Члени комісії:

 Лихвар Н. О.

 Кириченко О. В.



ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи та організації освітнього процесу ВНТУ

Олександр ПЕТРОВ
04 _____ 2026 р.

АКТ

Впровадження результатів дисертаційної роботи
*«Інформаційна технологія на основі спайкінгових нейронних мереж
для класифікації сигналів ЕКГ»*
Милосердова Дмитра Андрійовича
у освітній процес

Комісія у складі декана факультету інтелектуальних інформаційних технологій та автоматизації (ФІТА), к.т.н., доц. Севастьянова В. М., заступника декана ФІТА з навчально-методичної роботи, ст. викладача каф. системного аналізу та інформаційних технологій Присяжнюка В. В., завідувача кафедри комп'ютерних наук, д.т.н., проф. Ярового А. А., склали цей акт про те, що при вивченні освітніх компонентів «Сучасні парадигми штучних нейронних мереж» та «Нейрокомп'ютерні технології штучного інтелекту» підготовки здобувачів освітньо-професійної програми «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти, освітнього компоненту «Нейромережеві методи штучного інтелекту» підготовки здобувачів освітньо-професійної програми «Системи штучного інтелекту» за другим (магістерським) рівнем вищої освіти, та освітнього компоненту «Нейромережеві моделі та технології обчислювального інтелекту» здобувачів освітньо-професійної програми «Комп'ютерні науки» за третім (освітньо-науковим) рівнем вищої освіти спеціальності 122 «Комп'ютерні науки» впроваджено такі результати, розроблені Милосердовим Д. А.:

- інформаційну технологію класифікації сигналів ЕКГ на основі штучної спайкінгової нейронної мережі удосконаленої архітектури та спеціально розроблених методів навчання спайкінгової нейронної мережі, попередньої обробки та кодування сигналів ЕКГ, що має підвищену точність класифікації сигналів ЕКГ при збереженні ресурсоефективності та реально-часової швидкодії

- метод класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі, що дозволяє збільшити чутливість класифікації для класів, що відповідають аномаліям серцевого ритму;

- архітектуру спайкінгової нейронної мережі для використання у задачі класифікації сигналів ЕКГ, яка має за основу «резервуарні обчислення», 50 вхідних нейронів, що використовують Гаусові рецептивні поля для кодування сигналу ЕКГ, 2000 інтернейронів та по 1 вихідному нейрону на кожен клас сигналу ЕКГ.

Декан ФІТА

Володимир СЕВАСТЬЯНОВ

Заступник декана ФІТА

Василь ПРИСЯЖНЮК

Завідувач кафедри КН

Андрій ЯРОВИЙ

Додаток Б

Фрагмент лістингу програми

```
def gaussian_receptive_fields(x, n_neurons=20, min_val=0.0, max_val=1.0, sigma=None):
    """
    Encodes input x using Gaussian Receptive Fields.
    x: Input tensor [Batch, 1] or [Batch]
    Returns: [Batch, n_neurons]
    """
    if sigma is None:
        sigma = (max_val - min_val) / (n_neurons - 1)

    mu = torch.linspace(min_val, max_val, n_neurons).to(x.device)

    # Expand x to [Batch, 1] if needed
    if x.dim() == 1:
        x = x.unsqueeze(1)

    # x: [Batch, 1], mu: [n_neurons]
    # Broadcast subtraction: [Batch, n_neurons]
    return torch.exp(-((x - mu) ** 2) / (2 * sigma ** 2))

def init_weights_reservoir(module, mode='fixed_exc', sparsity=0.2, exc_ratio=0.8, w_scale_exc=1.0,
w_scale_inh=1.0):
    """
    Initializes weights with sparse connectivity and mixed Excitatory/Inhibitory synapses.
    Does NOT enforce Dale's Law on neurons (a single neuron can have both Exc and Inh synapses).

    Args:
    module: nn.Linear layer to initialize.
    mode: 'fixed_exc' (all positive) or 'mixed' (80/20 mix of +/-).
    sparsity: Probability of connection (0.2).
    exc_ratio: Ratio of excitatory connections in 'mixed' mode (0.8).
    w_scale_exc: Scaling factor for excitatory weights.
    w_scale_inh: Scaling factor for inhibitory weights.
    """
    with torch.no_grad():
        n_out, n_in = module.weight.shape

        # 1. Generate base random weights in range [0, 1]
        base_weights = torch.rand(n_out, n_in).to(device)

        weights = torch.zeros_like(base_weights)

        if mode == 'mixed':
            # Create boolean mask for Excitatory (True, ratio=exc_ratio) vs Inhibitory (False)
            is_exc = (torch.rand(n_out, n_in) < exc_ratio).to(device)

            # Apply scales and signs
            # Excitatory: [0, 1] * w_scale_exc * (+1)
            weights[is_exc] = base_weights[is_exc] * w_scale_exc

            # Inhibitory: [0, 1] * w_scale_inh * (-1)
            weights[~is_exc] = base_weights[~is_exc] * w_scale_inh * -1.0

        else:
            # Fixed Exc mode (all positive)
            weights = base_weights * w_scale_exc

        # 3. Apply Sparsity Mask
        # "Each input neuron connected with 20% of reservoir neurons"
        # statistically satisfied by global random mask with p=sparsity
        mask = (torch.rand(n_out, n_in) < sparsity).float().to(device)

        module.weight.data = weights * mask

    return mask

def plot_single_sample(model, loader):
    print("Plotting single sample...")
    model.eval()
    # Get one batch
    inputs, targets = next(iter(loader))
```

```

# Take the first sample
sample_input = inputs[0]
target_class = targets[0].item()

# Prepare Numpy version for plotting
if isinstance(sample_input, torch.Tensor):
    sample_input_np = sample_input.cpu().numpy()
else:
    sample_input_np = sample_input

# Prepare Tensor for Model
if isinstance(sample_input, np.ndarray):
    sample_input_tensor = torch.from_numpy(sample_input).float()
else:
    sample_input_tensor = sample_input.float()

# Ensure shape [Batch=1, Time, 1]
# If starting shape is [Time], make it [1, Time, 1]
if sample_input_tensor.dim() == 1:
    sample_input_tensor = sample_input_tensor.unsqueeze(0).unsqueeze(-1)
# If starting shape is [Time, 1], make it [1, Time, 1]
elif sample_input_tensor.dim() == 2:
    sample_input_tensor = sample_input_tensor.unsqueeze(0)

sample_input_tensor = sample_input_tensor.to(device)

print(f"[{datetime.now()}] Starting simulation for plotting...")
with torch.no_grad():
    input_spikes, lsm_spikes, out_spikes = model(sample_input_tensor, record=True)
print(f"[{datetime.now()}] Simulation completed.")
# Convert to numpy for plotting
input_spikes = input_spikes.squeeze(1).cpu().numpy()
lsm_spikes = lsm_spikes.squeeze(1).cpu().numpy()
# out_spikes is now logits [1, N_OUTPUT], not spikes over time. We skip plotting it as spikes.
out_logits = out_spikes.detach().cpu().numpy().flatten() # [N_OUTPUT]

# Plotting
fig, axs = plt.subplots(4, 1, figsize=(12, 12), sharex=False) # sharex=False because bar chart

# 1. Analog Input
axs[0].plot(sample_input_np)
axs[0].set_title(f"Analog Input Signal (Class {target_class})")

# 2. Input Neuron Spikes
spike_times, neuron_indices = np.where(input_spikes)
axs[1].scatter(spike_times, neuron_indices, s=10, c='k', marker='|')
axs[1].set_title("Input Neuron Spikes")

# 3. LSM Spikes
spike_times, neuron_indices = np.where(lsm_spikes)
axs[2].scatter(spike_times, neuron_indices, s=1, c='b', marker='.')
axs[2].set_title("Reservoir Spikes")

# 4. Output Logits (Bar Chart)
axs[3].bar(range(N_OUTPUT), out_logits)
axs[3].set_title("Output Logits (Prediction)")
axs[3].set_xticks(range(N_OUTPUT))
axs[3].set_xlabel("Class Index")
axs[3].axhline(0, color='k', linewidth=0.5)

plt.tight_layout()
plt.savefig('single_sample_response.png')
print("Saved single_sample_response.png")
plt.close()

def evaluate(model, test_loader):
    model.eval()
    correct = 0
    total = 0
    all_preds = []
    all_targets = []

    with torch.no_grad():
        for inputs, targets in test_loader:
            # Handle variable length inputs (same as in train)
            inputs_tensors = [torch.from_numpy(x).float() for x in inputs] if isinstance(inputs, list)
            else [inputs.float()] # Adjust if inputs is already tensor

```

```

# Since mit_bih.py now returns tensors in collate_fn if they are fixed size??
# actually custom_collate returns list of tensors or tensor
# If using standard DataLoader with stack it is tensor.
# Let's handle both for robustness
if isinstance(inputs, torch.Tensor):
    inputs_padded = inputs.to(device)
elif isinstance(inputs, list):
    # Add feature dim if needed
    inputs_tensors = [x.unsqueeze(-1) if x.dim() == 1 else x for x in inputs]
    inputs_padded = pad_sequence(inputs_tensors, batch_first=True).to(device)

targets = targets.long().to(device)

outputs = model(inputs_padded)
_, predicted = outputs.max(1)

total += targets.size(0)
correct += predicted.eq(targets).sum().item()

all_preds.extend(predicted.cpu().numpy())
all_targets.extend(targets.cpu().numpy())

acc = 100. * correct / total
recall = recall_score(all_targets, all_preds, average='macro', zero_division=0)
print(f"[{datetime.now()}] Test Accuracy: {acc:.2f}%, Macro Recall: {recall:.4f}")
print(classification_report(all_targets, all_preds, zero_division=0))
return recall

def plot_comprehensive_metrics(model, loader):
    print("Computing detailed metrics and plots...")
    model.eval()
    all_preds = []
    all_targets = []
    all_logits = []

    with torch.no_grad():
        for inputs, targets in loader:
            # Handle variable length inputs (same as in train)
            inputs_tensors = [torch.from_numpy(x).float() for x in inputs] if isinstance(inputs, list)
            else [inputs.float()]

            if isinstance(inputs, torch.Tensor):
                inputs_padded = inputs.to(device)
            elif isinstance(inputs, list):
                inputs_tensors = [x.unsqueeze(-1) if x.dim() == 1 else x for x in inputs]
                inputs_padded = pad_sequence(inputs_tensors, batch_first=True).to(device)

            targets = targets.long().to(device)
            outputs = model(inputs_padded)
            _, predicted = outputs.max(1)

            all_preds.extend(predicted.cpu().numpy())
            all_targets.extend(targets.cpu().numpy())
            all_logits.extend(outputs.cpu().numpy())

    all_targets = np.array(all_targets)
    all_preds = np.array(all_preds)
    all_logits = np.array(all_logits)
    all_probs = torch.softmax(torch.tensor(all_logits), dim=1).numpy()

    # 1. Kappa Score
    kappa = cohen_kappa_score(all_targets, all_preds)
    print(f"Cohen's Kappa Score: {kappa:.4f}")

    # 2. Confusion Matrix
    print("Plotting Confusion Matrix...")
    cm = confusion_matrix(all_targets, all_preds)
    # confusion_matrix plot
    try:
        fig, ax = plt.subplots(figsize=(12, 12))
        disp = ConfusionMatrixDisplay(confusion_matrix=cm)
        disp.plot(cmap='Blues', ax=ax, values_format='d')
        plt.title(f"Confusion Matrix (Kappa={kappa:.4f})")
        plt.savefig('confusion_matrix.png')
        print("Saved confusion_matrix.png")
        # plt.show()

```

```

except Exception as e:
    print(f"Could not plot confusion matrix: {e}")

# 3. ROC Curves (One-vs-Rest)
print("Plotting ROC Curves...")
try:
    n_classes = all_probs.shape[1]
    # Binarize targets for ROC
    y_test_bin = label_binarize(all_targets, classes=range(n_classes))

    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    plt.figure(figsize=(12, 8))
    colors = cycle(['blue', 'red', 'green', 'orange', 'purple', 'brown', 'pink', 'gray', 'olive',
'cyan', 'magenta', 'yellow'])

    for i, color in zip(range(n_classes), colors):
        # Only if class is present in test set (it should be)
        if np.sum(y_test_bin[:, i]) > 0:
            fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], all_probs[:, i])
            roc_auc[i] = auc(fpr[i], tpr[i])
            plt.plot(fpr[i], tpr[i], color=color, lw=2,
                    label='Class {0} (AUC = {1:0.2f})'.format(i, roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--', lw=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curves per Class')
    plt.legend(loc="lower right")
    plt.grid(True)
    plt.savefig('roc_curves.png')
    print("Saved roc_curves.png")
    # plt.show()
except Exception as e:
    print(f"Could not plot ROC curves: {e}")

def train_readout_correlation(model, loader):
    print(f"[{datetime.now()}] Starting Supervised Correlation Training (Reservoir -> Readout)...")
    model.eval()

    # Store accumulators
    # [N_OUTPUT, N_LSM] - Sum of spikes of each reservoir neuron for each class
    class_lsm_spikes_sum = torch.zeros(N_OUTPUT, N_LSM).to(device)
    class_samples_count = torch.zeros(N_OUTPUT).to(device)

    samples_processed = 0

    with torch.no_grad():
        for inputs, targets in loader:

            # Prepare inputs
            if isinstance(inputs, list):
                inputs_tensors = [torch.from_numpy(x).float() for x in inputs]
                inputs_tensors = [x.unsqueeze(-1) if x.dim() == 1 else x for x in inputs_tensors]
                inputs_padded = pad_sequence(inputs_tensors, batch_first=True).to(device)
            elif isinstance(inputs, torch.Tensor):
                inputs_padded = inputs.float().to(device)
                if inputs_padded.dim() == 2:
                    inputs_padded = inputs_padded.unsqueeze(-1)
            else:
                inputs_padded = torch.from_numpy(inputs).float().to(device)
                if inputs_padded.dim() == 2:
                    inputs_padded = inputs_padded.unsqueeze(-1)

            targets = targets.long().to(device)

            # Run model with record=True to capture LSM activity
            # Returns: (input_spikes, lsm_spikes, output_spikes)
            # lsm_spikes: [Time, Batch, N_LSM]
            _, lsm_spikes_t, _ = model(inputs_padded, record=True)

            # Sum over time -> [Batch, N_LSM]
            lsm_activity = lsm_spikes_t.sum(dim=0)

```

```

        # Accumulate per class
        for i in range(len(targets)):
            label = targets[i].item()
            class_lsm_spikes_sum[label] += lsm_activity[i]
            class_samples_count[label] += 1

        samples_processed += len(inputs)

# Calculate Average Spikes per Class for each LSM neuron
# [N_OUTPUT, N_LSM]
class_avg_lsm_spikes = torch.zeros_like(class_lsm_spikes_sum)
for c in range(N_OUTPUT):
    if class_samples_count[c] > 0:
        class_avg_lsm_spikes[c] = class_lsm_spikes_sum[c] / class_samples_count[c]

# --- Z-Score Normalization ---
# Convert activity to "deviations from the neuron's mean".
# This helps equalize "loud" and "quiet" neurons, and highlights specific activation.

# Mean and Std across classes [1, N_LSM]
mean_activity = class_avg_lsm_spikes.mean(dim=0, keepdim=True)
std_activity = class_avg_lsm_spikes.std(dim=0, keepdim=True)

# Avoid division by zero
epsilon = 1e-6
# Add a small noise floor relative to mean to avoid amplifying random noise in silence
noise_floor = mean_activity * 0.05 + epsilon

z_scores = (class_avg_lsm_spikes - mean_activity) / (std_activity + noise_floor)

# Use Z-Scores as the "correlation" metric
raw_correlations = z_scores

# --- Apply Constraints: Sparsity (20%) and E/I Ratio (80/20) ---

# We want to keep only top 20% connections for each output neuron.
# Out of those 20%, we want 80% to be Excitatory (most positive correlations)
# and 20% to be Inhibitory (most negative correlations).

final_weights = torch.zeros_like(raw_correlations)

sparsity = 0.2
exc_ratio = 1
total_connections = int(N_LSM * sparsity) # 200 connections
n_exc = int(total_connections * exc_ratio) # 160
n_inh = total_connections - n_exc          # 40

print(f"Applying constraints per class: {total_connections} total connections ({n_exc} Exc, {n_inh} Inh)")

for c in range(N_OUTPUT):
    class_corrs = raw_correlations[c] # [N_LSM]

    # 1. Select Excitatory (Top n_exc most positive values)
    # We only consider neurons that actually have positive correlation
    positive_indices = (class_corrs > 0).nonzero(as_tuple=True)[0]
    if len(positive_indices) > n_exc:
        # Get values for positive indices
        pos_values = class_corrs[positive_indices]
        # Find top k
        _, top_k_indices = torch.topk(pos_values, n_exc)
        # Map back to original indices
        exc_indices = positive_indices[top_k_indices]
    else:
        # Take all available if fewer than needed
        exc_indices = positive_indices

    # 2. Select Inhibitory (Top n_inh most negative values)
    # We only consider neurons that actually have negative correlation
    negative_indices = (class_corrs < 0).nonzero(as_tuple=True)[0]
    if len(negative_indices) > n_inh:
        # Get values (we want most negative, so topk on negated values or smallest)
        neg_values = class_corrs[negative_indices]
        # topk largest negative magnitude (smallest values)
        # largest=False gives smallest values
        _, bottom_k_indices = torch.topk(neg_values, n_inh, largest=False)

```

```

        inh_indices = negative_indices[bottom_k_indices]
    else:
        inh_indices = negative_indices

    # 3. Assign Weights
    # Normalizing to ranges [0, 1] for Exc and [-1, 0] for Inh
    exc_values = class_corrs[exc_indices]
    inh_values = class_corrs[inh_indices]

    if exc_values.numel() > 0:
        max_exc = exc_values.max()
        if max_exc > 0:
            exc_values = exc_values / max_exc

    if inh_values.numel() > 0:
        min_inh = inh_values.min()
        if min_inh < 0:
            inh_values = inh_values / min_inh.abs()

    # --- Priority Bias for Class 0 ---
    # Class 0 (Normal) typically has lower firing rates or less distinct features.
    # To improve its Recall (reduce False Alarms for Arrhythmia), we boost its weights saturation.
    if c == 0:
        exc_values = exc_values * 1.5
        exc_values = torch.clamp(exc_values, max=1.0)

    final_weights[c, exc_indices] = exc_values
    final_weights[c, inh_indices] = inh_values

    # Analyze generated weights
    print(f"\nWeight Statistics:")
    print(f"  Max Exc Weight: {final_weights.max().item():.4f}")
    print(f"  Max Inh Weight: {final_weights.min().item():.4f}")

    # Check "Specialists" count
    for c in range(N_OUTPUT):
        n_e = (final_weights[c] > 0).sum().item()
        n_i = (final_weights[c] < 0).sum().item()
        print(f"  Class {c}: {n_e} Exc, {n_i} Inh connections")

    # Apply weights
    model.fc_lsm_out.weight.data = final_weights

    print(f"\n[{datetime.now()}] Readout Weights Calculated & Updated.\n")

    save_model(model)

def train_readout_gradient(model, train_loader, test_loader, epochs=1, lr=1e-2):
    print(f"[{datetime.now()}] Starting Full Gradient Training (Input + Reservoir + Readout)...")

    criterion = nn.CrossEntropyLoss()

    # Optimizer for all parameters
    # Note: Reservoir training is slow/memory heavy but powerful.
    optimizer = torch.optim.AdamW(model.parameters(), lr=lr, weight_decay=1e-4) # AdamW usually better

    # Scheduler
    scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=epochs, eta_min=1e-5)

    best_recall = 0.0

    for epoch in range(epochs):
        model.train()

        running_loss = 0.0
        correct = 0
        total = 0

        # Iterate
        for i, (inputs, targets) in enumerate(train_loader):
            # Input Handling (same as evaluate/correlation)
            if isinstance(inputs, list):
                inputs_tensors = [torch.from_numpy(x).float() for x in inputs]
                inputs_tensors = [x.unsqueeze(-1) if x.dim() == 1 else x for x in inputs_tensors]
                inputs_padded = pad_sequence(inputs_tensors, batch_first=True).to(device)
            elif isinstance(inputs, torch.Tensor):
                inputs_padded = inputs.float().to(device)

```

```

        if inputs_padded.dim() == 2:
            inputs_padded = inputs_padded.unsqueeze(-1)
        else:
            inputs_padded = torch.from_numpy(inputs).float().to(device)
            if inputs_padded.dim() == 2:
                inputs_padded = inputs_padded.unsqueeze(-1)

        targets = targets.long().to(device)

        optimizer.zero_grad()

        # Forward: returns Logits [Batch, N_Output]
        outputs = model(inputs_padded)

        loss = criterion(outputs, targets)
        loss.backward()

        # Gradient Clipping (Important for RNN/SNN)
        nn.utils.clip_grad_norm_(model.parameters(), 1.0)

        optimizer.step()

        running_loss += loss.item()
        _, predicted = outputs.max(1)
        total += targets.size(0)
        correct += predicted.eq(targets).sum().item()

        if i % 100 == 0:
            print(f" [Epoch {epoch+1}/{epochs}] Step {i}, Loss: {loss.item():.4f}, Acc: {100.*correct/total:.2f}%")

            scheduler.step()
            epoch_loss = running_loss / len(train_loader)
            epoch_acc = 100. * correct / total
            print(f"[{datetime.now()}] Epoch {epoch+1} Finished using {device}. Loss: {epoch_loss:.4f}, Train Acc: {epoch_acc:.2f}%, LR: {scheduler.get_last_lr()[0]:.6f}")

            # Evaluate on Test set after each epoch
            # (Only every few epochs if slow)
            print("Evaluating on Test Set...")
            current_recall = evaluate(model, test_loader)

            # Save model if improvement
            if current_recall > best_recall:
                best_recall = current_recall
                print(f"New best recall: {best_recall:.4f}. Saving model...")
                save_model(model, "lsm_snn_trained.pth")

    print(f"Gradient Training Completed. Best Recall: {best_recall:.4f}")

def create_model(path="lsm_snn_model.pth"):
    print("Creating new model...")
    model = LSM_SNN().to(device)
    save_model(model, path)
    return model

def save_model(model, path="lsm_snn_model.pth"):
    print(f"Saving model to {path}...")
    torch.save(model.state_dict(), path)
    print("Model saved.")

def load_model(path="lsm_snn_model.pth"):
    if not os.path.exists(path):
        print(f"Model file {path} not found.")
        return create_model(path), True # is_new = True

    print(f"Loading model from {path}...")
    try:
        model = LSM_SNN().to(device)
        state_dict = torch.load(path, map_location=device)
        model.load_state_dict(state_dict)
        model.eval()
        print("Model loaded.")
        return model, False # is_new = False
    except Exception as e:
        print(f"Error loading model: {e}")
        print("Creating new model instead (overwrite incompatible file)...")

```

```

        return create_model(path), True

def save_weights_to_file(model, filename="snn_weights.npz"):
    print(f"Saving weights to {filename}...")
    w_in, w_rec, w_out = model.get_weights()
    np.savez(filename, input_weights=w_in, recurrent_weights=w_rec, output_weights=w_out)
    print(f"Weights saved to {filename}")

def save_weights_to_csv(model, folder="weights_csv"):
    if not os.path.exists(folder):
        os.makedirs(folder)

    print(f"Saving weights to CSV in folder '{folder}'...")
    w_in, w_rec, w_out = model.get_weights()

    # Save with 4 decimal places
    np.savetxt(os.path.join(folder, "input_weights.csv"), w_in, delimiter=",", fmt="%.4f")
    np.savetxt(os.path.join(folder, "recurrent_weights.csv"), w_rec, delimiter=",", fmt="%.4f")
    np.savetxt(os.path.join(folder, "output_weights.csv"), w_out, delimiter=",", fmt="%.4f")

    print(f"Weights saved to CSV files in '{folder}'/.")

def plot_pr_curves(y_true, y_scores, n_classes):
    """
    Plots Precision-Recall curves for each class.
    y_true: Ground truth labels (1D)
    y_scores: Raw logits or probabilities [Batch, n_classes]
    """
    # Binarize labels
    Y = label_binarize(y_true, classes=range(n_classes))

    precision = dict()
    recall = dict()
    average_precision = dict()

    for i in range(n_classes):
        precision[i], recall[i], _ = precision_recall_curve(Y[:, i], y_scores[:, i])
        average_precision[i] = average_precision_score(Y[:, i], y_scores[:, i])

    # A "micro-average": quantifying score on all classes jointly
    precision["micro"], recall["micro"], _ = precision_recall_curve(Y.ravel(), y_scores.ravel())
    average_precision["micro"] = average_precision_score(Y, y_scores, average="micro")

    # Plot
    colors = cycle(['navy', 'turquoise', 'darkorange', 'cornflowerblue', 'teal'])

    plt.figure(figsize=(10, 8))
    f_scores = np.linspace(0.2, 0.8, num=4)
    lines = []
    labels = []

    # Plot iso-f1 curves
    for f_score in f_scores:
        x = np.linspace(0.01, 1)
        y = f_score * x / (2 * x - f_score)
        (l,) = plt.plot(x[y >= 0], y[y >= 0], color='gray', alpha=0.2)
        plt.annotate('f1={0:0.1f}'.format(f_score), xy=(0.9, y[45] + 0.02))

    plt.plot(recall["micro"], precision["micro"], color='gold', lw=2,
             label='Micro-average PR curve (area = {0:0.2f})'.format(average_precision["micro"]))

    for i, color in zip(range(n_classes), colors):
        plt.plot(recall[i], precision[i], color=color, lw=1.5,
                 label='Class {0} (area = {1:0.2f})'.format(i, average_precision[i]))

    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Precision-Recall Curve per Class')
    plt.legend(loc="lower left", fontsize='small')
    plt.savefig('pr_curves.png')
    print("Saved pr_curves.png")
    plt.close()

def analyze_reservoir_separability(model, loader, samples=500):
    """

```

```

Visualizes the separation of classes in the high-dimensional reservoir state using t-SNE.
Collects the mean firing rates of the reservoir neurons for a subset of data.
"""
print("Collecting reservoir states for t-SNE...")
model.eval()

reservoir_states = []
labels = []
count = 0

with torch.no_grad():
    for inputs, targets in loader:
        if count >= samples:
            break

        inputs = inputs.to(device)
        # Ensure input is [Batch, Time, 1]
        if inputs.dim() == 2:
            inputs = inputs.unsqueeze(-1)

        # Forward pass with recording to get reservoir spikes
        # Note: We need to modify forward or use a hook, but here we can
        # just use the public forward method if it returns what we need.
        # The current forward() returns logits. We need to access the accumulation inside.
        # To avoid changing forward() just for this, we'll manually run part of it
        # or rely on `record=True` returning (in_spikes, lsm_spikes, logits).

        _, lsm_spikes_t, _ = model(inputs, record=True)
        # lsm_spikes_t: [Time, Batch, N_LSM]

        # Calculate mean firing rate per sample over time: [Batch, N_LSM]
        mean_rates = lsm_spikes_t.mean(dim=0)

        reservoir_states.append(mean_rates.cpu().numpy())
        labels.append(targets.numpy())

        count += inputs.size(0)

reservoir_states = np.concatenate(reservoir_states, axis=0)[:samples]
labels = np.concatenate(labels, axis=0)[:samples]

print(f"Running t-SNE on {reservoir_states.shape[0]} samples with {reservoir_states.shape[1]}
dimensions...")
tsne = TSNE(n_components=2, random_state=42, perplexity=30, init='pca', learning_rate='auto')
X_embedded = tsne.fit_transform(reservoir_states)

plt.figure(figsize=(10, 8))
sns.scatterplot(x=X_embedded[:,0], y=X_embedded[:,1], hue=labels, palette="tab10", legend='full',
alpha=0.7)
plt.title("t-SNE Visualization of Reservoir Liquid States")
plt.xlabel("Dim 1")
plt.ylabel("Dim 2")
plt.legend(title="Class", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.savefig('reservoir_tsne.png')
print("Saved reservoir_tsne.png")
plt.close()

def analyze_activity_stats(model, loader, samples=200):
    """
    Calculates and plots statistics about network activity:
    - Mean Firing Rate (Hz)
    - Sparsity (Silent Neurons)
    - Activity Distribution
    """
    print("Analyzing network activity statistics...")
    model.eval()

    all_lsm_spikes = []

    count = 0
    with torch.no_grad():
        for inputs, _ in loader:
            if count >= samples:
                break
            inputs = inputs.to(device)
            if inputs.dim() == 2:

```

```

        inputs = inputs.unsqueeze(-1)

        _, lsm_spikes_t, _ = model(inputs, record=True)
        # lsm_spikes_t: [Time, Batch, N_LSM]

        all_lsm_spikes.append(lsm_spikes_t.cpu())
        count += inputs.size(0)

# Concatenate along Batch dim: from list of [T, B, N] -> one huge tensor?
# Actually just keeping track of statistics is enough to avoid OOM.

total_spikes = 0
total_neurons = N_LSM
total_time_steps = 0
total_elements = 0

neuron_activity_sum = torch.zeros(N_LSM)

for batch_spikes in all_lsm_spikes:
    # batch_spikes: [T, B, N]
    T, B, N = batch_spikes.shape

    sum_per_neuron = batch_spikes.sum(dim=(0, 1)) # Sum over Time and Batch
    neuron_activity_sum += sum_per_neuron

    total_spikes += batch_spikes.sum().item()
    total_time_steps += T * B
    total_elements += T * B * N

# Mean Firing Rate
# Total Spikes / (Total Time * Total Neurons) * (1000 / DT) if DT is in ms?
# Here inputs are steps.
# MFR (Hz) = (Total Spikes / Total Elements) * (1000 / DT)

avg_firing_rate = (total_spikes / total_elements) * (1000 / DT)

# Sparsity: % of neurons that never fired across these samples ??
# Or usually, sparsity is "fraction of zeros in the matrix".
# We can calculate "Population Sparsity" (per timestep) or "Lifetime Sparsity".
# Let's do simple population sparsity: 1 - ratio of active elements
population_sparsity = 1.0 - (total_spikes / total_elements)

# Silent Neurons: Neurons that didn't fire at all in this dataset subset
silent_neurons = (neuron_activity_sum == 0).sum().item()
percent_silent = (silent_neurons / total_neurons) * 100

print(f"--- Activity Stats ---")
print(f"Mean Firing Rate: {avg_firing_rate:.2f} Hz")
print(f"Population Sparsity: {population_sparsity*100:.2f}% (Avg inactive neurons per step)")
print(f"Silent Neurons: {silent_neurons} ({percent_silent:.2f}% - Neurons that never fired)")

# Plot Activity Histogram
# Normalize activity by time (Hz per neuron)
neuron_hz = (neuron_activity_sum / total_time_steps) * (1000 / DT)

plt.figure(figsize=(10, 5))
plt.hist(neuron_hz.numpy(), bins=50, color='purple', alpha=0.7)
plt.title("Distribution of Neuron Firing Rates")
plt.xlabel("Firing Rate (Hz)")
plt.ylabel("Count of Neurons")
plt.yscale('log') # Log scale often helpful for sparse networks
plt.grid(True, which="both", ls="-", alpha=0.2)
plt.savefig('activity_stats.png')
print("Saved activity_stats.png")
plt.close()

def estimate_synaptic_operations(model, loader, samples=100):
    """
    Estimates the number of Synaptic Operations (SynOps).
    Accumulates total spikes from input and reservoir layers.
    Multiplies by fan-out (number of outgoing connections).
    """
    print("Estimating Energy (Synaptic Operations)...")
    model.eval()

    w_in_lsm, w_lsm_lsm, w_lsm_out = model.get_weights()

```

```

# Count non-zero connections (fan-out sum)
# w_in_lsm: [N_INPUT, N_LSM] -> Input spikes trigger N_LSM potential ops?
# Actually, sparse connectivity means we count non-zeros.
# But usually SOP = sum(spikes_i * fan_out_i)

# Column sum of adjacency matrix gives in-degree. Row sum gives out-degree.
# Weights shapes from get_weights(): [Source, Target]

# Fan-out per neuron
fan_out_input = np.count_nonzero(w_in_lsm, axis=1) # [N_INPUT]

# Handle potentially fused weights in recurrent layer (Input+Hidden)
# w_lsm_lsm is [Source, Target].
# If source dim is > N_LSM, the latter part handles recurrent spikes.
if w_lsm_lsm.shape[0] > N_LSM:
    # Assuming standard [Input, Hidden] concatenation
    # We take the slice corresponding to hidden state feedback
    w_lsm_recurrent = w_lsm_lsm[w_lsm_lsm.shape[0] - N_LSM:, :]
    fan_out_lsm = np.count_nonzero(w_lsm_recurrent, axis=1) # [N_LSM]
else:
    fan_out_lsm = np.count_nonzero(w_lsm_lsm, axis=1) # [N_LSM]

# We also have LSM -> Output
# This is a dense layer (Linear), but rate-coded input.
# However, for energy estimation in neuromorphic hardware, if the readout is also spiking
# or accumulated, every LSM spike triggers an update to output accumulators.
fan_out_lsm_to_readout = np.count_nonzero(w_lsm_out, axis=1)

total_lsm_fan_out = fan_out_lsm + fan_out_lsm_to_readout

total_synops = 0
total_samples = 0

with torch.no_grad():
    for inputs, _ in loader:
        if total_samples >= samples:
            break

        inputs = inputs.to(device)
        if inputs.dim() == 2:
            inputs = inputs.unsqueeze(-1)

        in_spikes_t, lsm_spikes_t, _ = model(inputs, record=True)
        # Shapes: [Time, Batch, Neurons]

        # Sum spikes over time and batch
        in_spike_counts = in_spikes_t.sum(dim=(0, 1)).cpu().numpy() # [N_INPUT]
        lsm_spike_counts = lsm_spikes_t.sum(dim=(0, 1)).cpu().numpy() # [N_LSM]

        # SOP = Spikes * FanOut
        batch_synops_in = np.dot(in_spike_counts, fan_out_input)
        batch_synops_lsm = np.dot(lsm_spike_counts, total_lsm_fan_out)

        total_synops += (batch_synops_in + batch_synops_lsm)
        total_samples += inputs.size(0)

avg_synops_per_sample = total_synops / total_samples
print(f"--- Energy Estimation ---")
print(f"Avg Synaptic Operations per inference: {avg_synops_per_sample:,.0f} FLOPs/SOPs")
print(f" (Assuming 1 Spike -> Mult/Add ops = FanOut)")

if __name__ == "__main__":
    MODEL_PATH = "lsm_snn_model_rr.pth"

    # 1. Download/Load Data
    print("Downloading/Loading data...")
    train_loader, test_loader = download_mitbih(balance_data=True)

    # 2. Create or Load Model
    model, is_new_model = load_model(MODEL_PATH)

    # 3. Training (if new or requested)
    # We'll validate if we need to train.
    # For now, let's assume we want to evaluate the current model,
    # but if it's new, we must train.
    if is_new_model:
        print("Model is new. Starting training...")

```

```

    train_readout_gradient(model, train_loader, test_loader, epochs=70, lr=1e-3)
    save_model(model, MODEL_PATH)
else:
    print("Model loaded. Skipping training (set is_new_model=True to force retrain).")

# 4. Standard Evaluation
print("\n=== Standard Evaluation ===")
evaluate(model, test_loader)

plot_comprehensive_metrics(model, test_loader)

# 5. Advanced Analysis for Article/Dissertation
print("\n=== Running Advanced Characterization ===")

# A. Reservoir Separability (t-SNE)
# Shows how well the liquid separates classes in high-dim space
analyze_reservoir_separability(model, test_loader, samples=1000)

# B. Network Activity Stats
# Proves sparsity and efficiency
analyze_activity_stats(model, test_loader, samples=500)

# C. Precision-Recall Curves
# Better than ROC for imbalanced data
print("Computing Precision-Recall Curves...")
model.eval()
all_targets = []
all_probs = []
with torch.no_grad():
    for inputs, targets in test_loader:
        if isinstance(inputs, list):
            inputs_tensors = [torch.from_numpy(x).float() for x in inputs]
            inputs_tensors = [x.unsqueeze(-1) if x.dim() == 1 else x for x in inputs_tensors]
            inputs_padded = pad_sequence(inputs_tensors, batch_first=True).to(device)
        elif isinstance(inputs, torch.Tensor):
            inputs = inputs.to(device)
            inputs_padded = inputs.unsqueeze(-1) if inputs.dim() == 2 else inputs
        else:
            inputs_padded = torch.from_numpy(inputs).float().to(device)
            if inputs_padded.dim() == 2:
                inputs_padded = inputs_padded.unsqueeze(-1)

        targets = targets.long().to(device)
        logits = model(inputs_padded)
        probs = torch.softmax(logits, dim=1)

        all_targets.extend(targets.cpu().numpy())
        all_probs.extend(probs.cpu().numpy())

plot_pr_curves(np.array(all_targets), np.array(all_probs), n_classes=N_OUTPUT)

# D. Plot Single Sample Response
# Visual proof of spiking behavior
plot_single_sample(model, test_loader)

# Save weights for other analyses
save_weights_to_csv(model)

import wfdb
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
import torch
from torch.utils.data import DataLoader, TensorDataset
from scipy.interpolate import interp1d
import os
from datetime import datetime

def min_max_normalization(arr):
    t_min = np.min(arr)
    t_max = np.max(arr)
    if t_max == t_min:
        return np.zeros_like(arr)
    return (arr - t_min) / (t_max - t_min)

def trim(arr):

```

```

num_elements_to_remove = int(len(arr) * 0.2)
new_arr = arr[num_elements_to_remove:-num_elements_to_remove]
return new_arr

class VariableSizeDataset(torch.utils.data.Dataset):
    def __init__(self, data, targets):
        self.data_tensor = torch.tensor(data, dtype=torch.float32)
        # Add channel dimension: [N, Time] -> [N, Time, 1]
        if self.data_tensor.ndim == 2:
            self.data_tensor = self.data_tensor.unsqueeze(-1)
        self.targets = targets

    def __len__(self):
        return len(self.data_tensor)

    def __getitem__(self, idx):
        return self.data_tensor[idx], self.targets[idx]

def download_mitbih(balance_data=True, window_size=130) -> tuple[DataLoader, DataLoader]:
    """Download the MIT-BIH Arrhythmia Database if not already present."""
    # Define the class mapping (AAMI classes: 0-N, 1-S, 2-V, 3-F, 4-Q)
    class_map = {
        'N': 0, # Normal
        'L': 1, # Normal
        'R': 2, # Normal
        'e': 3, # Normal
        'j': 4, # Normal

        'A': 5, # Supraventricular
        'a': 6, # Supraventricular
        'J': 7, # Supraventricular
        'S': 8, # Supraventricular

        'V': 9, # Ventricular
        'E': 10, # Ventricular

        'F': 11, # Fusion
        # '/': 4, 'f': 4, 'Q': 4 # Unknown/Paced/Q
    }

    # List of MIT-BIH records (excluding paced ones: 102,104,107,217)
    records = [
        100, 101, 103, 105, 106, 108, 109, 111, 112, 113, 114, 115, 116, 117, 118, 119,
        121, 122, 123, 124, 200, 201, 202, 203, 205, 207, 208, 209, 210, 212, 213, 214,
        215, 219, 220, 221, 222, 223, 228, 230, 231, 232, 233, 234
    ]

    # Download the MIT-BIH database if not already present
    db_path = 'mitdb'
    if not os.path.exists(db_path):
        wfdb.dl_database('mitdb', db_path)

    target_len = 200
    cache_path = os.path.join(db_path, f'processed_cache_dynamic_centered_{target_len}.npz')

    if os.path.exists(cache_path):
        print(f"[{datetime.now()}] Loading data from cache: {cache_path}")
        data = np.load(cache_path, allow_pickle=True)
        all_beats = data['beats']
        all_labels = data['labels']
    else:
        # Collect all beats and labels
        all_beats = []
        all_labels = []

        rr1_list = []
        rr2_list = []

        print(f"[{datetime.now()}] Processing records with dynamic window (target {target_len})...")

        for rec in records:
            # Load signal and annotations
            sig, fields = wfdb.rdsamp(f'{db_path}/{rec}')
            ann = wfdb.rdann(f'{db_path}/{rec}', 'atr')
            signal = sig[:, 0]

            beat_indices = [i for i, s in enumerate(ann.symbol) if s in class_map]

```

```

for k in range(1, len(beat_indices) - 1):
    i = beat_indices[k]
    prev_i = beat_indices[k-1]
    next_i = beat_indices[k+1]

    pos = ann.sample[i]
    prev_pos = ann.sample[prev_i]
    next_pos = ann.sample[next_i]

    start = (prev_pos + pos) // 2
    end = (pos + next_pos) // 2

    rr1_list.append(pos - start)
    rr2_list.append(end - pos)

    if start < 0 or end > len(signal):
        continue

    # Split so R-peak is at center (start of right half)
    beat_left = signal[start:pos]
    beat_right = signal[pos:end]

    beat_left = stretch_to_length(beat_left, target_len // 2)
    beat_right = stretch_to_length(beat_right, target_len // 2)

    beat = np.concatenate((beat_left, beat_right))

    # Min-Max Normalization per beat
    beat = min_max_normalization(beat)

    all_beats.append(beat)
    all_labels.append(class_map[ann.symbol[i]])

print(f"[{datetime.now()}] Finished processing records.")

if len(rr1_list) > 0:
    print(f"Average RR1 (prev midpoint to peak): {np.mean(rr1_list):.2f} samples")
    print(f"Average RR2 (peak to next midpoint): {np.mean(rr2_list):.2f} samples")

# Save as standard float array since length is fixed
all_beats = np.array(all_beats, dtype=np.float32)
all_labels = np.array(all_labels, dtype=np.int64)

np.savez(cache_path, beats=all_beats, labels=all_labels)
print(f"[{datetime.now()}] Data saved to cache.")

print(f"[{datetime.now()}] Total beats: {len(all_beats)}")
print(f"[{datetime.now()}] Label distribution: {np.bincount(all_labels)}")

if balance_data:
    # Balance the dataset to 2500 samples per class
    X_balanced = []
    y_balanced = []

    unique_classes = np.unique(all_labels)
    target_count = 2500

    print(f"[{datetime.now()}] Resampling to {target_count} per class...")

    for cls in unique_classes:
        cls_indices = np.where(all_labels == cls)[0]
        cls_beats = all_beats[cls_indices]
        cls_labels = all_labels[cls_indices]

        # Enable replacement only if we possess fewer samples than requested
        replace = len(cls_beats) < target_count

        X_resampled, y_resampled = resample(
            cls_beats, cls_labels,
            replace=replace,
            n_samples=target_count,
            random_state=42
        )
        X_balanced.append(X_resampled)
        y_balanced.append(y_resampled)

```

```

    all_beats = np.vstack(X_balanced)
    all_labels = np.concatenate(y_balanced)

    print(f"[{datetime.now()}] Balanced label distribution: {np.bincount(all_labels)}")

# Stratified split into train/test (80/20)
X_train, X_test, y_train, y_test = train_test_split(
    all_beats, all_labels, train_size=0.8, stratify=all_labels, random_state=42
)

# Create Datasets
# using simple Tensor dataset now that size is fixed
y_train = torch.tensor(y_train, dtype=torch.long)
y_test = torch.tensor(y_test, dtype=torch.long)

train_dataset = VariableSizeDataset(X_train, y_train)
test_dataset = VariableSizeDataset(X_test, y_test)

# Create DataLoaders
batch_size = 32
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

print(f"[{datetime.now()}] Train beats: {len(X_train)}, Test beats: {len(X_test)}")
return train_loader, test_loader

def stretch_to_length(arr, target_len=200):
    if len(arr) < 2:
        return np.zeros(target_len)
    x_old = np.linspace(0.0, 1.0, len(arr))
    x_new = np.linspace(0.0, 1.0, target_len)
    f = interp1d(x_old, arr, kind='linear', fill_value="extrapolate")
    return f(x_new)

```

Додаток В

Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації

[1] Милосердов Д. А., Колесницький О. К., Волосович О. В., Жумагулова Ш., Короленко О. В. Класифікація електрокардіограм як динамічного процесу на основі спайкінгової нейронної мережі. Оптико-електронні інформаційно-енергетичні технології. 2024. Вип. 48, № 2. С. 68–77. DOI: 10.31649/1681-7893-2024-48-2-68-77.

[2] Милосердов Д. А., Колесницький О. К. Застосування спайкінгових нейронних мереж до класифікації електрокардіограм. Огляд. Наука і техніка сьогодні. Серія «Фізико-математичні науки». 2025. № 9(50). С. 1612–1643. DOI: 10.52058/2786-6025-2025-9(50)-1612-1643.

[3] Милосердов Д. А., Колесницький О. К. Метод попередньої обробки сигналу ЕКГ для класифікації електрокардіограм за допомогою спайкінгових нейронних мереж. Технічні науки та технології. 2025. № 3(41). С. 203–210. DOI: 10.25140/2411-5363-2025-3(41)-203-210. URL: <http://tst.stu.cn.ua/article/view/345202/332545>.

[4] Милосердов Д. А., Колесницький О. К. Метод класифікації електрокардіограм із застосуванням спайкінгових нейронних мереж. Вісник Вінницького політехнічного інституту. 2025. № 5. С. 121–128. DOI: 10.31649/1997-9266-2025-182-5-121-128

[5] Милосердов Д. А., Колесницький О. К. Інформаційна технологія класифікації сигналів ЕКГ на основі спайкінгової нейронної мережі. Наука і техніка сьогодні. Серія «Фізико-математичні науки». 2026. № 2(56). С. 2535–2552. DOI: 10.52058/2786-6025-2026-2(56)-2535-2552.

[6] Семчук В. О., Милосердов Д. А., Колесницький О. К. Інформаційна технологія розпізнавання вимовлених літер на основі спайкінгової нейронної мережі. Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)». Вінниця, 2023. [Електронний ресурс]. URL:

<https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/16854/1405>
5.

[7] Милосердов Д. А., Колесницький О. К. Розпізнавання серцевої недостатності як багатопараметричного процесу на основі спайкінгових нейронних мереж. Матеріали 1-ої Міжнародної науково-практичної конференції «Інформаційні системи та технології: результати і перспективи (IST 2024)», 6 березня 2024 р. Київ : ФІТ КНУТШ, 2024. [Електронний ресурс]. URL:

https://ist.fit.knu.ua/_files/ugd/016074_8fc56ecbbfa94cde874eedd872f3780e.pdf.

[8] Милосердов Д. А., Колесницький О. К. Метод попередньої обробки сигналу електрокардіограм для класифікації за допомогою спайкінгових нейронних мереж. Science, Technology and Industry in the Digital Age : Collection of Scientific Papers with Proceedings of the 1st International Scientific and Practical Conference. International Scientific Unity. September 24–26, 2025. Hamburg, Germany. P. 159–164. DOI: 10.70286/isu-24.09.2025.

[9] Милосердов Д. А., Колесницький О. К. Інформаційна технологія класифікації сигналів ЕКГ за допомогою спайкінгових нейронних мереж. Innovative Research in Science and Economy : Collection of Scientific Papers with Proceedings of the 4th International Scientific and Practical Conference. International Scientific Unity. Brussels, Belgium. April 1–3, 2026. P. 178–181. DOI: 10.70286/ISU-01.04.2026.009.

[10] Свідоцтво про реєстрацію авторського права на твір № 139885. Комп'ютерна програма «Інформаційна технологія класифікації часових рядів». Колесницький О. К., Денисов І. К., Милосердов Д. А. Дата реєстрації: 06.10.2025.