КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки

на тему:

КЛАСИФІКАЦІЯ ЕКГ СИГНАЛІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Виконала студентка 4 курсу
Харченко Вікторія Віталіївна _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Панченко Тарас Володимирович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захистуна засіданні кафедри теорії та технології програмування
«01» червня 2022 р.,
протокол № 10

Завідувач кафедри
М. С. Нікітченко _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 56 сторінок, 29 ілюстрацій, 11 таблиць, 12 джерел посилань.

КЛАСИФІКАЦІЯ ЕКГ, ФІЛЬТРАЦІЯ, ЛОКАЛІЗАЦІЯ R ПІКІВ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ , НАЇВНИЙ БАЄСІВ КЛАСИФІКАТОР, ЛОГІСТИЧНА РЕГРЕСІЯ, ВИПАДКОВІ ДЕРЕВА, МЕТОД ОПОРНИХ ВЕКТОРІВ.

Об’єктом дослідження є цифровий ЕКГ сигнал, призначений для виявлення проблем з електричною активністю серця та графічного відтворення показників роботи серця.

Метою роботи є розробка методу автоматизованої діагностики для виявлення аритмій та інших серцевих хвороб.

Методи дослідження: фільтрація ЕКГ, реалізація алгоритму виявлення R-піків, класифікація ЕКГ глибокими нейронними мережами та методами машинного навчання. Інструменти розробки: середовище розробки Visual Studio 2019, мова програмування C++, середовище інтерактивних обчислень Jupyter Notebook, мова програмування Python.

Результати роботи: проведено огляд та аналіз існуючих підходів для аналізу та класифікації ЕКГ сигналів; реалізовано фільтр рухомого середнього та фільтр Баттерворта для зменшення шуму в кардіограмах; реалізовано алгоритм визначення R-піків, навчено 5 нейронних мереж для класифікації ЕКГ; проведено класифікацію, базуючись на результатах нейронних мереж, з використанням таких алгоритмів машинного навчання як наївний баєсів класифікатор, випадкові дерева, метод опорних векторів та логістична регресія.

Результатом роботи є розроблений метод класифікації ЕКГ сигналів, який показав конкурентні результати в порівнянні з існуючими підходами.

Найкращих результатів вдалося досягнути в визначенні нормального синусового ритму та миготливої аритмії.

Розроблений підхід для класифікації в подальшому може бути використаний на вбудованих системах при тривалих кардіологічних спостереженнях для вчасного виявлення деяких “мовчазних” серцевих захворювань. Також реалізований алгоритм виявлення R-піків разом з алгоритмами фільтрації може використовуватись для контролю частоти серцевих скорочень під час активної спортивної діяльності.

Значимість роботи полягає у можливості проведення точної діагностики серцевих захворювань, здатної своєчасно попередити прогресування та наслідки хвороби.

Подальші дослідження можуть стосуватися покращення існуючого рішення, розробки алгоритмів локалізації та аналізу P і T хвиль. Важливим кроком є інтеграція моделі у вбудовані системи.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Серцево-судинні захворювання вже кілька десятків років є однією з головних причин смертності людей у всьому світі. Діагностика цієї групи захворювань насамперед починається з проведення достатньо простої неінвазивної процедури – електрокардіографії, яка є тестом призначеним для виявлення проблем з електричною активністю серця та графічного відтворення показників роботи серця. Дуже важливою частиною електрокардіографії є розшифровка результатів запису, яка зазвичай проводиться лікарем на підставі його медичного досвіду. Відповідно до цього дуже важливою стає точність і правильність попередньої оцінки та постановки діагнозу лікарем, що може бути значно спрощено за допомогою автоматизації даного процесу, впровадженням технологій машинного навчання та штучного інтелекту, що сприяє виключенню можливої помилки у діагностиці, спричиненої «людським фактором».

На сьогоднішній день запропоновано декілька підходів щоб полегшити це завдання та скоротити час аналізу. Навіть без абсолютної точності ці підходи дозволяють швидко виділити підозрілі сегменти кардіограми для поглибленого аналізу фахівцем. Запропоновані рішення поділяються на два типи. Методи першого типу (медичні алгоритми) моделюють логіку лікаря-діагноста. Це можуть бути алгоритми, що знаходять довжину найважливіших елементів ЕКГ і порівнюють їх показники з нормою. Методи другого типу (немедичні алгоритми) базуються на застосуванні методів багатовимірного статистичного аналізу та теорії ймовірності. До немедичного аналізу відноситься використання алгоритмів машинного навчання. В даному випадку, розробник не обов’язково знає і розуміє, що відбувається всередині системи.

Актуальність роботи та підстави для її виконання. На жаль, серцеві захворювання часто мають мовчазний характер, де епізоди можуть траплятися нечасто. В цьому випадку виникає потреба в реєстрації записів серцевої діяльності протягом декількох днів, які в свою чергу вимагають довгого та трудомісткого аналізу кваліфікованим електрофізіологом. Прикладом такої хвороби є фібриляція передсердь (ФП) або миготлива аритмія, що характеризується некоординованою електричною активністю передсердь. Дослідження показують, що цей тип аритмії збільшує ризик появи інсульту в 5 разів. Ситуація ускладнюється проблемою раннього розпізнавання, що, у свою чергу, затримує лікування, здатне вчасно попередити прогресування захворювання.

Саме тому проблема своєчасної та правильної діагностики тяжких порушень ритму серця належить до актуальних завдань кардіології, а потреба в якісному автоматизованому аналізі знятої ЕКГ є нагальною.

Мета і завдання роботи. Метою даної роботи є вивчення існуючих рішень для аналізу серцево-судинних захворювань та розробка власної автоматизованої діагностичної системи для виявлення аритмій та інших серцевих хвороб. Для її досягнення була поставлена низка задач:

- проведення огляду та аналізу існуючих алгоритмів локалізації R-піків та класифікації серцевих захворювань на основі ЕКГ сигналів;
- дослідження та застосування відомих алгоритмів фільтрації з метою покращення якості сигналу;
- розробка та порівняння різних методів машинного навчання для класифікації кардіограм.

Об’єкт, предмет і методи дослідження, засоби розробки. Об’єктом дослідження виступає цифровий ЕКГ сигнал. Предметом дослідження є

ефективні методи аналізу електрокардіограм. Попереднє очищення даних від шуму і вирівнювання сигналу було реалізовано за допомогою відомих алгоритмів фільтрації. Це було зроблено для підвищення точності локалізації R-піків та класифікації кардіограм. Безпосередньо для класифікації було розроблено 5 згорткових нейронних мереж, а також використано відомі алгоритми машинного навчання для вибору остаточного класу.

Для навчання нейронних мереж використовувався датасет *PhysioNet/Computing in Cardiology Challenge-2017*. Більшість алгоритмів було реалізовано в інтегрованому середовищі розробки Microsoft Visual Studio 2019 мовою C++. Для розробки архітектури та тестування нейронних мереж та інших методів машинного навчання використовувалось середовище Jupyter Notebook і мова програмування Python.

Можливі сфери застосування. Розроблений метод в поєднанні з пристроєм зняття ЕКГ може бути використано для тривалих безперервних кардіологічних спостережень, а саме для поточного контролю частоти серцевих скорочень (ЧСС), що є одним з основних гемодинамічних показників серцево-судинної системи, та для розпізнавання порушень серцевого ритму. Крім того, рішення може бути використане для моніторингу серця під час виконання спортивних вправ.

РОЗДІЛ 1 ПОСТАНОВКА ТА ДОСЛІДЖЕННЯ ЗАДАЧІ

1.1 Опис даних

Електрокардіографія - досить простий неінвазивний метод реєстрації та дослідження електричних полів, що утворюються при роботі серця. Стандартна кардіограма включає в себе 12 відведень, по кожному з яких будується графік. Рисунки 1.1 та 1.2 показують графік нормального серцевого циклу здорової людини.

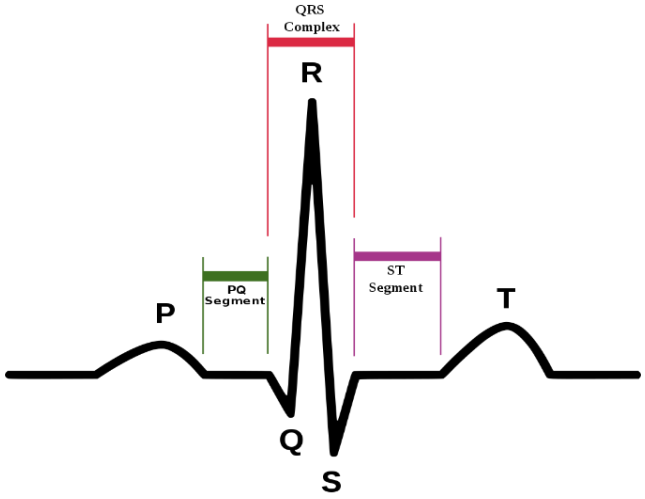


Рисунок 1.1 – Морфологія здорового удару серця на ЕКГ

На електрокардіограмі розрізняють такі основні компоненти, що відповідають різним фазам серцевого циклу:

- P-зубець (P-хвиля) відповідає фазі деполяризації (скорочення) передсердь;
- QRS-комплекс (шлуночковий комплекс) відповідає фазі деполяризації шлуночків;
- ST-сегмент, що відповідає періоду між закінченням деполяризації та початком реполяризації шлуночків;
- T-зубець (T-хвиля) відповідає фазі реполяризації шлуночків.

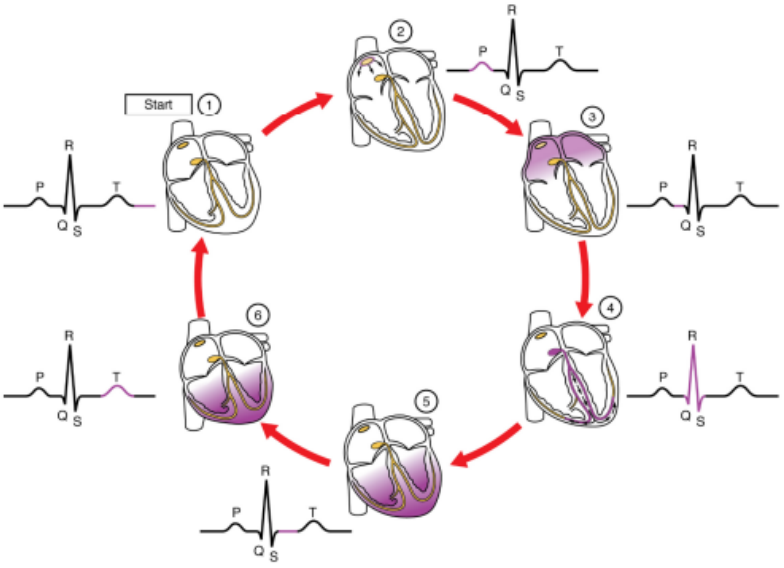


Рисунок 1.2 – Нормальний серцевий цикл

У приладах та системах кардіологічного спостереження найбільша увага приділяється аналізу серцевого ритму та його порушенням шляхом відстеження динамічних змін в електрокардіограмі під час її тривалого спостереження. Це передбачає безперервну реєстрацію та аналіз кардіосигналу протягом великої кількості серцевих циклів. При цьому основою автоматичного аналізу є вимірювання інтервалів часу між послідовними скороченнями шлуночків серця (RR-інтервалів) та аналіз характеру форми основних хвиль кардіоциклу.

В даній дипломній роботі основну увагу було зосереджено на локалізацію R-піків та подальшу класифікацію ЕКГ сигналу за допомогою нейронних мереж. Для цього використовувались ЕКГ з ресурсу *PhysioNet* [1] з датасету *Computing in Cardiology Challenge-2017* [2]. Записи ЕКГ у наборі даних були зібрані за допомогою пристрою AliveCor з частотою 300 Гц. Датасет містить в собі 8528 однополюсних записів ЕКГ-сигналів, кожен з яких позначений одним з чотирьох класів: нормальний синусовий ритм, миготлива аритмія, інший вид ритму або зашумлена кардіограма, що позначалися

відповідно як "N", "AF", "O" та "~". Відсоткове співвідношення кожного класу можна побачити в таблиці 1.1. На рисунку 1.3 показана візуалізація кардіограм з датасету.

Таблиця 1.1 – Кількісне співвідношення ЕКГ різних класів в [2]

Клас	N	AF	O	~
К-сть	5050	738	2456	284
Відсоток	59.2%	8.6%	28.8%	3.4%

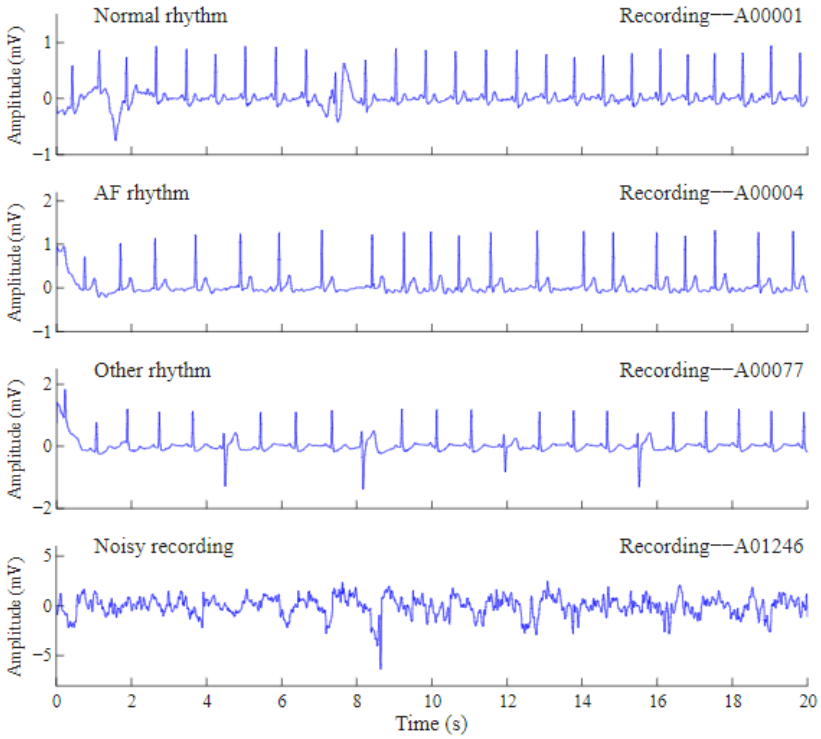


Рисунок 1.3 – Приклади ЕКГ з кожного з чотирьох класів

Як можна побачити зі зразків ЕКГ, визначальною рисою аритмії є зміна традиційної P-хвилі на f-хвилі низьких амплітуд та нерегулярність серцевого ритму. В свою чергу, клас Other може включати в себе різні серцеві порушення, тому він важче піддається узагальненню та класифікації.

1.2 Огляд задачі автоматизованого аналізу ЕКГ

Тема автоматизованого аналізу ЕКГ та застосування методів машинного навчання для діагностики захворювань серця досить часто зустрічається у новітніх наукових дослідженнях. Основні задачі, що найчастіше описуються в роботах з аналізу ЕКГ, відображені на рисунку 1.4. Вони діляться на 5 типів: діагностика захворювань, розпізнавання QRS-комплексу, дослідження фаз сну, ідентифікація особистості людини за даними ЕКГ, очищення від шуму. Дані з ЕКГ можуть бути отримані як зі спеціального медичного апарату, так і з будь-якого «розумного» пристрою (годинник, фітнес-браслет тощо).

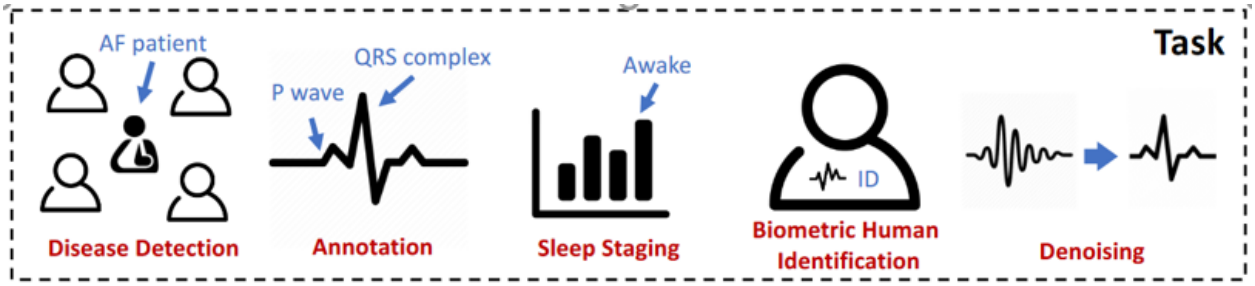


Рисунок 1.4 – Популярні задачі аналізу ЕКГ

В свою чергу автоматичний аналіз ЕКГ у системах кардіологічного спостереження характеризується низкою специфічних особливостей, а саме:

- складність та різноманітність сигналу, що проявляється у наявності індивідуальних особливостей у кожного пацієнта, пов'язаних як з типом конституції людини, так і з присутністю тих чи інших патологій у провідній системі серця;
- мінливість сигналу в часі, яка може бути обумовлена як змінами положення тіла, так та фізіологічними феноменами - зміною загального стану пацієнта, розвитком патологій у серцево-судинній системі та появою порушень ритму серця;

- наявність різного роду шуму, що неминуче виникають у процесі тривалого спостереження сигналу;
- необхідність отримання результатів аналізу в режимі реального часу (або в прискореному режимі, як при аналізі холтерівських записів), що накладає обмеження на обчислювальну складність алгоритмів з точки зору завантаження процесора і обсягу оперативної пам'яті;
- використання обмеженої кількості відведень ЕКГ (у деяких застосуваннях — лише одного відведення), що ускладнює отримання точних оцінок діагностичних показників через недостатньо повну інформацію, що міститься у вхідному сигналі.

Як правило, процес автоматичного аналізу ЕКГ у системах кардіологічного спостереження включає наступні основні етапи:

1) попередня обробка сигналу;
2) виявлення QRS-комплексу ЕКГ та вимірювання RR-інтервалу;
3) розпізнавання порушень серцевого ритму;

До першого етапу відносяться такі задачі як усунення чи послаблення шуму, що піддається приглушенню, та оцінка рівня шуму у сигналі з метою виключення фрагментів, подальший аналіз який неможливий внаслідок їхньої високої зашумленості. На другому етапі вирішуються задачі виділення інформативних складових сигналу з метою забезпечення найкращих умов наступних етапів обробки, виявлення QRS-комплексу на фоні інших елементів кардіоциклу ЕКГ та визначення меж комплексу й умовної опорної точки, що служить для вимірювання RR-інтервалу. На третьому етапі формуються висновки відносно характеру ритму серця.

Схожий підхід має модель класифікації ЕКГ розроблена в рамках даної дипломної роботи.

1.3 Огляд розробленого рішення

В даній дипломній роботі було реалізовано новий підхід до класифікації ЕКГ сигналу. В основі рішення лежать 5 згорткових нейронних (CNN) мереж, одна для виявлення зашумлених ЕКГ (P), три для бінарних та одна для тернарної класифікації на такі класи: нормальний синусовий ритм (N), миготлива аритмія (A), інший вид ритму (O). Перед подачею даних нейронним мережам вони проходять попередню обробку, до якої відноситься застосування фільтрів для усунення шуму і вирівнювання основної лінії та подальше виявлення R-піків.

Таким чином на першому кроці відбувається відокремлення зашумлених кардіограми від інших за допомогою нейронної мережі. Основною складністю такої класифікації є те, що деякі кардіограми можуть одночасно суміщати кілька типів, наприклад, одна частина ЕКГ нормальна, а інша зашумлена. Далі застосовується рухоме середнє і фільтра Баттерворта, за чим слідує алгоритм локалізації R піків, розрізання ЕКГ на частини по кілька RR циклів та їх передискретизація. На останньому кроці виконується класифікація кожної частини, використовуючи бінарні та тернарні CNN, які видають ймовірності належності до того чи іншого класу. В цій роботі було досліджено декілька варіантів вибору кінцевого класу, базуючись на отриманих ймовірностях, а саме: використовуючи зважений максимум ймовірностей, вибір за допомогою наївного баєсівого класифікатора, логістичної регресії, випадкових дерев та методу опорних векторів.

Для оцінки точності класифікації була використана запропонована в [2] $\boldsymbol{F_1}$ метрика:

$$\boldsymbol{F_{1n} = \frac{2\,Nn}{\Sigma N + \Sigma n}} \quad \boldsymbol{F_{1a} = \frac{2\,Aa}{\Sigma A + \Sigma a}} \quad \boldsymbol{F_{1o} = \frac{2\,Oo}{\Sigma O + \Sigma o}} \quad \boldsymbol{F_{1p} = \frac{2\,Pp}{\Sigma P + \Sigma p}}$$

$$\boldsymbol{F_1 = \frac{F_{1n} + F_{1a} + F_{1o}}{3}},$$

де N, A, O, P – загальна кількість ЕКГ, n, a, o, p – кількість класифікованих ЕКГ, Nn, Aa, Oo, Pp – кількість правильно визначених ЕКГ.

Таблиця підрахунку результату зображена на рисунку 1.5.

TABLE II
COUNTING RULES

		Predicted Class				
		Normal	AF	Other	Noisy	Total
Reference Class	Normal	Nn	Na	No	Np	∑N
	AF	An	Aa	Ao	Ap	∑A
	Other	On	Oa	Oo	Op	∑O
	Noisy	Pn	Pa	Po	Pp	∑P
	Total	∑n	∑a	∑o	∑p	

Рисунок 1.5 – Таблиця для підрахунку результатів

РОЗДІЛ 2 ПОПЕРЕДНЯ ОБРОБКА КАРДІОГРАМ

2.1 Зменшення шуму в ЕКГ

В умовах тривалого безперервного знімання ЕКГ сигналу неминучою є поява в сигналі шуму. Шум можна визначити як будь-який сигнал, який не є пов’язаним з електричною активністю серця. Це, в свою чергу, ускладнює аналіз сигналу та вимірювання його параметрів.

Шум може мати багато можливих джерел. Він може бути спричиненим втратою контакту між електродом і шкірою (контактний шум електродів), м'язовими скороченнями, рухом (короткочасне зміщення електрода), диханням (додавання до ЕКГ синусоїдальної складової з частотою дихання) та апаратним шумом, що генерується електричними пристроями знімання та посилення сигналів. На рисунку 2.1 можна побачити шум в ЕКГ, спричинений втратою контакту між шкірою та електродом, та на рисунку 2.2 зображене так зване «блукання», що викликане скороченням та розслабленням м’язів грудної клітки.

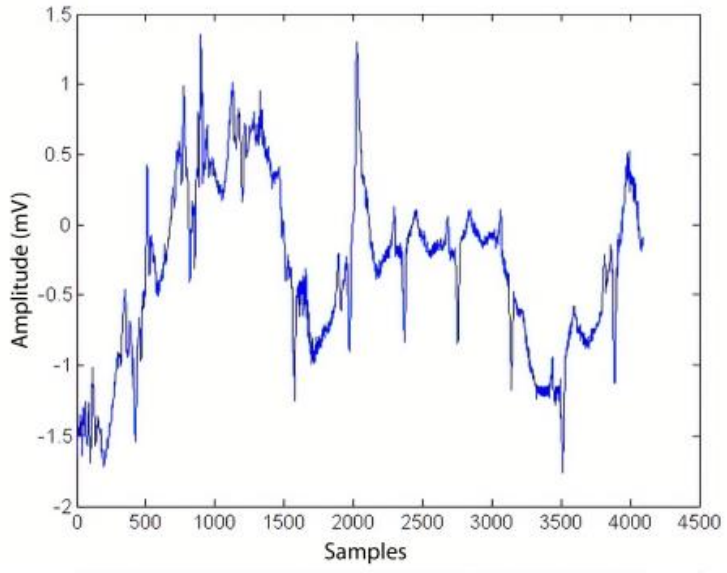


Рисунок 2.1 – Шум, спричинений рухом електрода

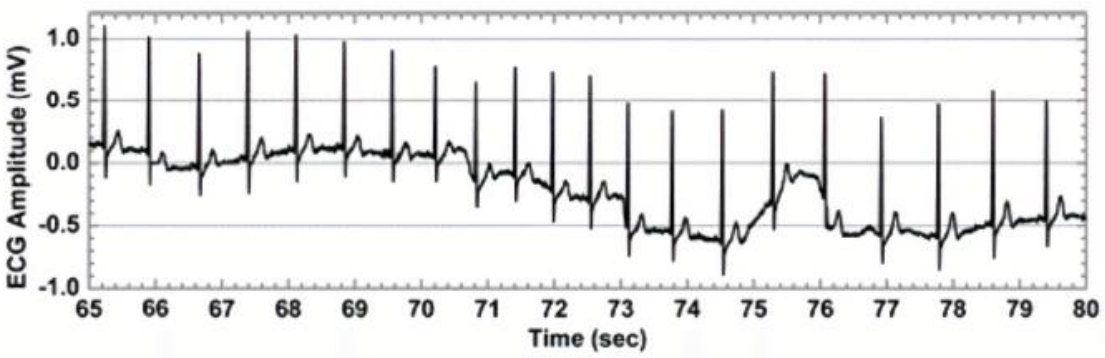


Рисунок 2.2 – «Блукання» базової лінії

Різноманітність шуму та його непередбачуваність суттєво ускладнюють автоматичний аналіз ЕКГ сигналу. У зв'язку з цим алгоритми обробки ЕКГ повинні передбачати спеціальні заходи, що забезпечують їх стійкість до шуму. На сьогоднішній день існує багато рішень для роботи с біопотенціалами, які включають в себе навіть не один, а цілу систему фільтрів. Однак їх використання повинно бути виправданим, так як фільтри часто вносять досить серйозні спотворення у форму сигналу. Так, наприклад, фільтри високих частот, що використовуються для стабілізації ізоелектричної лінії, призводять до спотворення сегмента ST ЕКГ. А фільтри низьких частот, які призначені для фільтрації шуму, зменшуються амплітуду біосигнала.

В рамках цієї дипломної роботи було реалізовано фільтр рухомого середнього, який використовувався для усунення коливань основної лінії, та низькочастотний фільтр Баттерворта для фільтрації високочастотного шуму.

2.1.1 Реалізація рухомого середнього

Нехай $\boldsymbol{x_n}$- вхідний сигнал фільтра, $\boldsymbol{y_n}$ - вихідний сигнал, **P** - порядок фільтра. Тоді рівняння, що характеризує фільтр рухомого середнього матиме вигляд :

$$\boldsymbol{y_n = \frac{1}{P+1}\sum_{i=0}^{P} x_{n-i}}$$

Представивши $\boldsymbol{R}$ як радіус рухомого середнього, рівняння можна подати наступною формулою:

$$\boldsymbol{f_A[n] = \frac{1}{2R+1} \sum_{k=-R}^{R} x[n+k]}$$

В розробленому рішенні виконувалось дворазове застосування рухомого середнього. Перше застосування має малий радіус та сприяє згладжуванню QRS комплексу. Друге використання має достатньо великий радіус і призначене для знаходження основної лінії. В рамках цієї роботи було досліджено різні варіанти радіусів та використано найкращий виявлений варіант - радіус 0.1 і 0.3 секунди для першого і другого кроку відповідно. Отримане рухоме середнє віднімається від вхідної ЕКГ, що можна представити наступною формулою:

$$\boldsymbol{y_A[n] = x[n] - f_A[n], f_A = MA(MA(x, R = 30), R = 90)}$$

Результат роботи фільтру рухомого середнього можна побачити на рисунку 2.3.

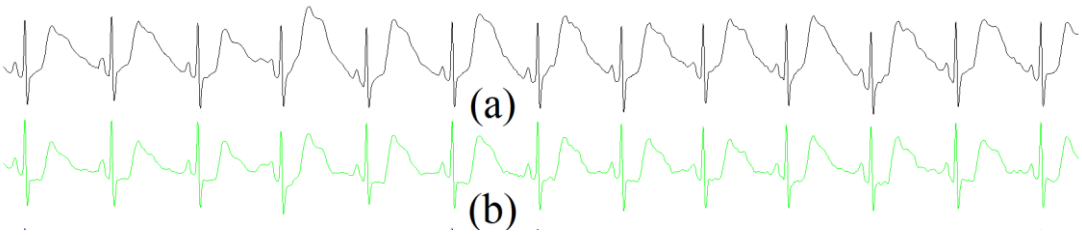


Рисунок 2.3 – ЕКГ після усунення основної лінії: a – вхідна ЕКГ;
b – застосування рухомого середнього

На даному рисунку можна побачити, що фільтр добре справляється з короткими і різкими перепадами в сигналі та в той же час не спотворює морфологію PQRST, що є надзвичайно важливим для подальшої класифікації кардіограми нейронними мережами.

2.1.2 Реалізація фільтра Баттерворта

Передавальна функція фільтра Баттерворта може бути записана наступною формулою:

$$\mathbf{H(s)} = \frac{\mathbf{Y(s)}}{\mathbf{X(s)}} = \frac{\mathbf{G_0}}{\boldsymbol{B_n(\alpha)}} = \frac{\mathbf{1}}{\boldsymbol{B_n(\alpha)}}, \boldsymbol{\alpha} = \frac{\boldsymbol{s}}{\boldsymbol{w_c}},$$

де **n-** порядок, $\boldsymbol{w_c}$- частота зрізу, $\boldsymbol{B_n(\alpha)}$ це нормалізований поліном Баттерворта.

$$\boldsymbol{B_n(s)} = \begin{cases} \prod_{k=1}^{\frac{n}{2}} \left(s^2 - 2s\cos\left(\frac{2k+n-1}{2n}\pi\right) + 1\right), & n = 2m \\ (s+1)\prod_{k=1}^{\frac{n-1}{2}} \left(s^2 - 2s\cos\left(\frac{2k+n-1}{2n}\pi\right) + 1\right), & n = 2m-1 \end{cases}$$

Щоб перетворити даний аналоговий фільтр в цифровий, можна використати білінійне перетворення з оберненням:

$$\boldsymbol{s} = \frac{\mathbf{1}}{\boldsymbol{T}}\mathbf{log}(\boldsymbol{z}) \approx \frac{\mathbf{2}}{\boldsymbol{T}}\frac{\boldsymbol{z}-\mathbf{1}}{\boldsymbol{z}+\mathbf{1}} = \frac{\mathbf{2}}{\boldsymbol{T}}\frac{\mathbf{1}-\boldsymbol{z}^{-\mathbf{1}}}{\mathbf{1}+\boldsymbol{z}^{-\mathbf{1}}}$$

Для уникнення зсуву частоти зрізу $\boldsymbol{w_c}$ застосуємо обгортання частоти зрізу, що матиме вигляд:

$$\boldsymbol{s} = \frac{\boldsymbol{w_c}}{\mathbf{tan}\left(\frac{\boldsymbol{w_c T}}{\mathbf{2}}\right)}\frac{\mathbf{1}-\boldsymbol{z}^{-\mathbf{1}}}{\mathbf{1}+\boldsymbol{z}^{-\mathbf{1}}}$$

Тоді отриманий цифровий фільтр можна представити наступним чином:

$$\mathbf{H(z)} = \mathbf{H}\left(\boldsymbol{s} = \frac{\boldsymbol{w_c}}{\mathbf{tan}\left(\frac{\boldsymbol{w_c T}}{\mathbf{2}}\right)}\frac{\mathbf{1}-\boldsymbol{z}^{-\mathbf{1}}}{\mathbf{1}+\boldsymbol{z}^{-\mathbf{1}}}\right) = \frac{\mathbf{Y(z)}}{\mathbf{X(z)}} = \frac{\sum_{k=0}^{n} \boldsymbol{b_k z^{-k}}}{\sum_{k=0}^{n} \boldsymbol{a_k z^{-k}}}$$

Переписавши в математичний вигляд, отримаємо кінцевий фільтр:

$$\boldsymbol{y[n] = \sum_{k=0}^{n} a_k\, x[n-k] - \sum_{k=1}^{n} b_k\, y[n-k]}$$

В даній дипломній роботі були використані наступні параметри для фільтрації ЕКГ: $\boldsymbol{w_c = 40\pi}$, $\boldsymbol{n = 12}$, $\boldsymbol{T = \frac{1}{300}}$. На рисунку 2.4 зображено результат застосування побудованого фільтру Баттерворта на зашумленій кардіограмі.

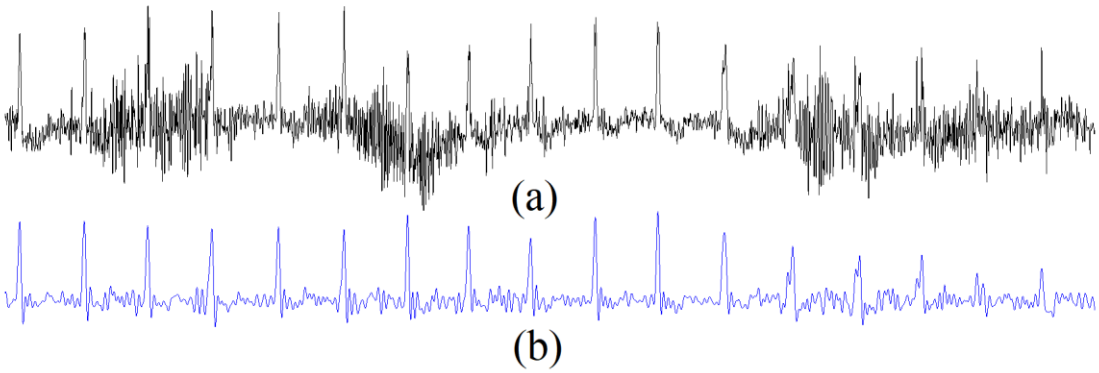


Рисунок 2.4 – Фрагмент ЕКГ після фільтрації високочастотного шуму: a – вхідна ЕКГ; b – відфільтрована ЕКГ

2.2 Локалізація R-піків

Аналіз QRS комплексу є однією з найважливіших частин аналізу ЕКГ-сигналу. Завдяки його якісному визначенню можна оцінити частоту серцевих скорочень та різниці в довжинах RR інтервалів.

В даній дипломній роботі було розроблено алгоритм знаходження R піків який складається з наступних кроків:

- трансформація сигналу, що базується на перепаді між локальними екстремумами;
- пошук найвищих точок ЕКГ та позначення потенційних R піків;
- групування потенційних R піків і вибір максимуму з кожної групи.

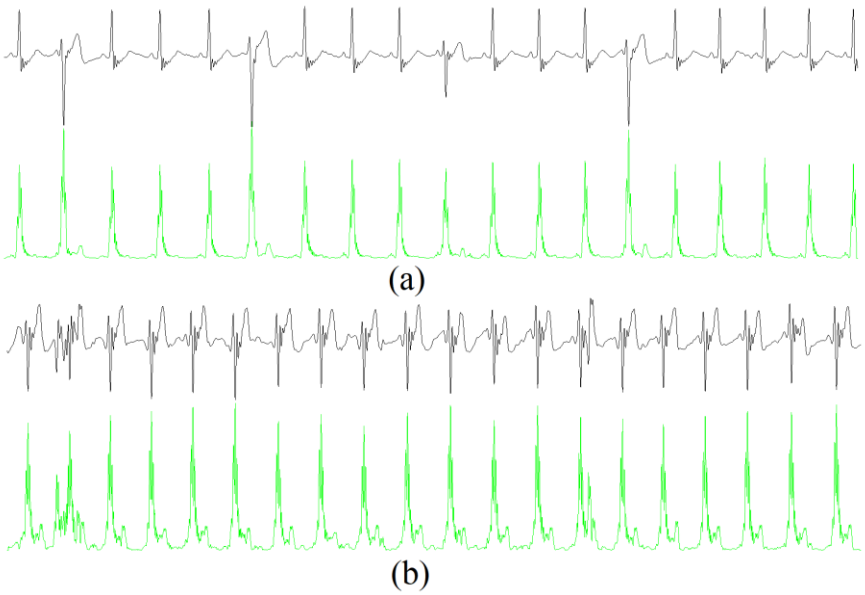
Головне завдання першого кроку полягає у збільшенні амплітуди R-піків та зменшенні амплітуди P і T хвиль. Цей крок допомагає виділити різкі зміни в амплітуді характерні лише для QRS комплексу. Також на даному кроці R піки, напрямлені вниз, перевертаються вгору. Алгоритм трансформації можна представити наступною формулою:

$$\boldsymbol{P1_k} = \left| \min_{j=k-HD,k-1} (x_j) - x_k \right| + \left| \min_{j=k+1,k+HD} (x_j) - x_k \right|$$

$$\boldsymbol{P2_k} = \left| \max_{j=k-HD,k-1} (x_j) - x_k \right| + \left| \max_{j=k+1,k+HD} (x_j) - x_k \right|$$

$$\boldsymbol{P_k^*} = \max(\boldsymbol{P1_k}, \boldsymbol{P2_k})$$

Тут $\boldsymbol{HD}$ (Half Duration) – деяка константа, що приблизно дорівнює половині довжини QRS комплексу. Проте, так як його тривалість може варіюватись в залежності від людини, береться усереднене значення $\boldsymbol{HD}$. В даній реалізації $\boldsymbol{HD}$ було взято рівним 0.03 В результаті трансформація збільшує ті значення, які мають короткий та різкий перепад амплітуди, що характерно для QRS комплексу, та не сильно впливає на більш плавні переходи, які характерні для P та T хвиль. Результат застосування трансформації показано на рисунку 2.5.



(a)

(b)

Рисунок 2.5 – Застосування трансформації. Чорним – відфільтрована ЕКГ, зеленим – трансформація: a – A00077 з [2]; b – A00084 з [2]

На другому етапі визначення R-піків розглядається частина ЕКГ, що потрапляє в вікно, яке в середньому захоплює кілька RR циклів здорової людини. Всі числа, що попадають у вікно, сортуються та з них обираються максимальні значення, які в свою чергу позначаються як потенційні R-піки. Гіпертрофовані P і T хвилі, разом з шумом, можуть сприйматись програмою як хибні R-піки. Для уникнення таких ситуацій виконується додаткове відсікання деяких визначених R зубців, а саме, якщо зліва або справа від визначеного R-піка є число, більше за дане хоча б в 2 рази, то дане число вважається хибнопозитивним R-піком і видаляється з вибірки.

Останнім етапом є групування потенційних R-піків в кластери та вибір максимального значення з кластеру. Два ймовірних R-піки потрапляють до одного кластеру, якщо знаходяться на відстані меншій ніж певна константа **D**. В даній реалізації **D** була рівна 0.25 секунди.

Результати визначення R-піків можна побачити на рисунку 2.6.

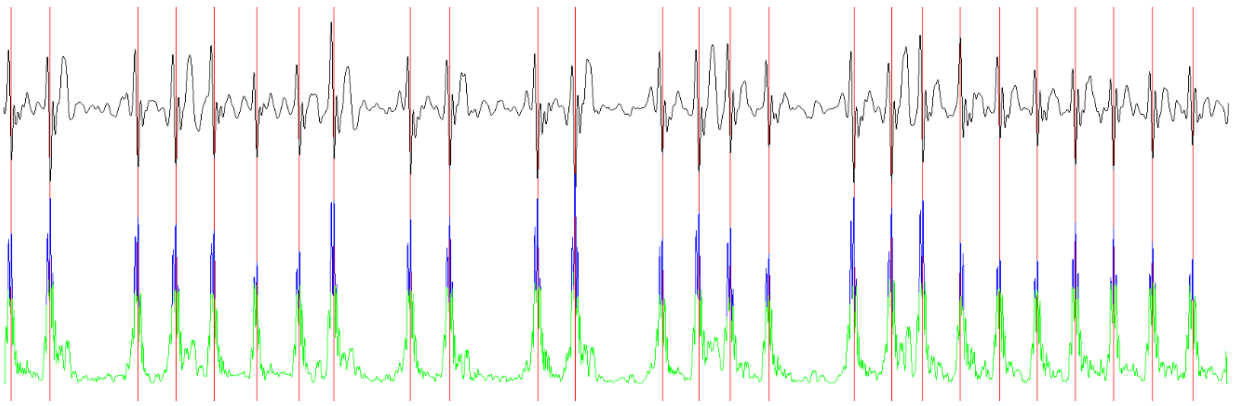


Рисунок 2.6 – Локалізація R-піків. Чорним – вхідна ЕКГ A00706 з [1], зеленим – трансформація, синім – потенційні R-піки, червоним – визначені R-піки

Для тестування алгоритму був використаний набір ЕКГ з MIT-BIH Arrhythmia Database [3], що містить розмічені R-піки. Основним показником якості алгоритму є $\boldsymbol{T_p}$ **–** кількість правильно визначених R піків, $\boldsymbol{F_p}$ – кількість хибнопозитивно визначених R піків, $\boldsymbol{F_n}$ – кількість хибнонегативно визначених R піків та похибка $\boldsymbol{F_d}$:

$$\boldsymbol{F_d = \frac{F_p + F_n}{T_p + F_p + F_n}}$$

В таблиці 2.1 показано результати порівняно з рішенням представленим в [4].

Таблиця 2.1 – Порівняння якості виявлення R-піків

Розроблений алгоритм			**[4]**		
$\boldsymbol{F_p}$	$\boldsymbol{F_n}$	$\boldsymbol{F_d}$	$\boldsymbol{F_p}$	$\boldsymbol{F_n}$	$\boldsymbol{F_d}$
40	**141**	**0.2256**	**244**	**192**	**0.5759**

Після визначення R-піків кардіограма розрізається по декілька RR циклів. Оскільки довжина циклів може кожного разу відрізняється, проводиться передискретизація виділених RR циклів так, щоб вихідна послідовність мала задану в нейронній мережі довжину.

РОЗДІЛ 3 ЗАСТОСУВАННЯ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ В ЗАДАЧІ КЛАСИФІКАЦІЇ ЕКГ

3.1 Огляд існуючих підходів

Незважаючи на велику кількість робіт, присвячених побудові ефективних алгоритмів автоматичної класифікації електрокардіограм, ця проблема, як і раніше, залишається актуальною. Моделі, засновані на вручну створених методах вилучення ознак, можуть не враховувати прихованих залежностей, необхідних для більш точної класифікації. Водночас моделі глибоких нейронних мереж не вимагають людських знань предметної області, в якій вони застосовуються, та здатні виділяти найважливіші закономірності в даних. Глибокі нейронні мережі були вперше використані для класифікації ЕКГ відносно недавно, але вони одразу ж показали перспективні результати, продемонструвавши високу якість розпізнавання деяких патологій. Тим не менш, такі моделі чутливі до типу вхідного сигналу та якості розмітки.

Широке поширення цифрових систем діагностики здоров'я призвело до накопичення великих масивів медичних даних і, як наслідок, до можливості покращення існуючих систем автоматичної діагностики за рахунок використання нейронних мереж. Для вирішення задач класифікації електрокардіограм були запропоновані різні архітектури глибоких нейронних мереж. Найчастіше використовуваними підходами для класифікації ЕКГ є використання рекурентних та згорткових нейронних мереж.

3.1.1 Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) – тип нейронних мереж, призначений для роботи з послідовностями. У RNN вихідні дані попереднього кроку використовуються як вхідні дані для наступного, таким чином, шляхом ітеративного оновлення станів вони здатні запам'ятовувати інформацію в послідовному порядку. Завдяки такій можливості RNN мережі

застосовуються, наприклад, в завданнях обробки природної мови, оскільки здатні знаходити взаємозв'язок між токенами і, таким чином, розуміють контекст. На рисунку 3.1 зображена схема рекурентної мережі.

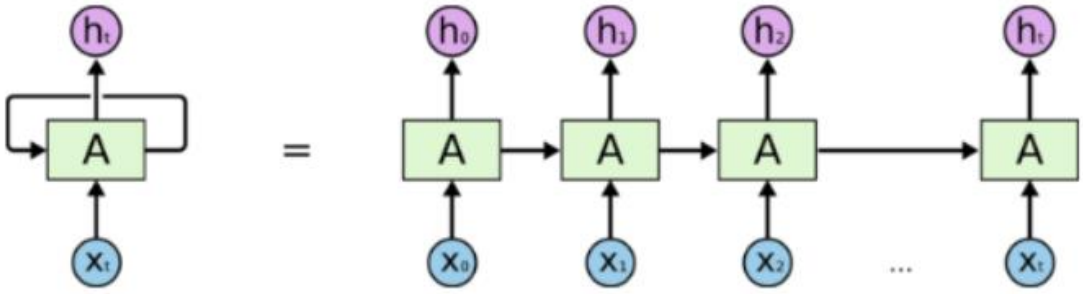


Рисунок 3.1 – Розгорнута рекурентна мережа

Дані ЕКГ, за своєю сутністю є числовими послідовностями, тому вибір рекурентних нейронних мереж є логічним і обґрунтованим як для знаходження тимчасових залежностей, так і для обробки вхідних даних різної довжини. Найчастіше можна побачити використання саме мережі довгої короткострокової пам'яті (LSTM) для аналізу ЕКГ. Так наприклад автори [5] розробили рекурентну LSTM мережу для класифікації окремих серцевий ударів, характерних для аритмії та нормального ритму. В свою чергу у статті [6] використовуються невеликі мережі LSTM, які вирішують завдання класифікації безперервно в режимі реального часу.

LSTM-мережа (Long short-term memory) - це особливий вид рекурентних нейронних мереж, що містить рекурентні LSTM-модулі здатні запам'ятовувати значення як на короткі, так і довгі проміжки часу. Причиною такої можливості є те, що LSTM-модуль не використовує функції активації всередині своїх рекурентних компонентів. Таким чином, значення, що зберігається, не розмивається в часі, і градієнт не зникає при використанні

методу зворотного розповсюдження помилки в часі при навчанні штучної нейронної мережі.

LSTM-блоки містять так звані «вентилі» (gates), які використовуються для контролю потоків інформації на входах та на виходах пам'яті даних блоків. Ці вентилі реалізовані в вигляді логістичної функції для обчислення значень в діапазоні [0; 1]. Множення на це значення використовується для часткового допуску чи заборони потоку інформації всередину і назовні пам'яті. Так, «вхідний вентиль» (input gate) контролює міру входження нових значень в пам'ять, а «вентиль забування» (forget gate) контролює міру збереження значень в пам'яті. «Вихідний вентиль» (output gate) контролює якою мірою значення, що знаходиться в пам'яті, використовується при розрахунку вихідної функції активації для блоку. Схему LSTM-мережі можна побачити на рисунку 3.2

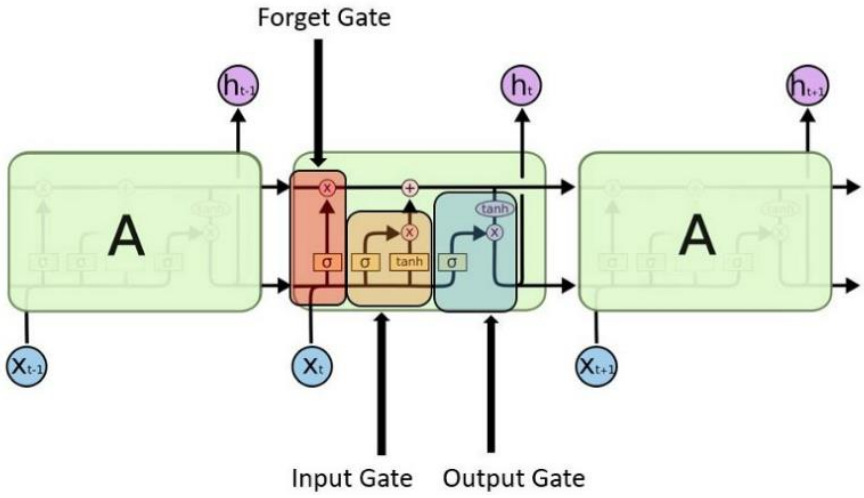


Рисунок 3.2 – Схема LSTM-мережі

3.1.2 Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) є класом нейронних мереж, що відмінно зарекомендували себе у вирішенні задачі класифікації зображень. Сучасне покоління популярного обладнання для глибокого навчання в

основному виконується на відеокартах Nvidia, які, у свою чергу, використовують ядра CUDA і оптимізовані для роботи з паралельними обчисленнями. На відміну від рекурентних нейронних мереж, у яких обчислення відбуваються послідовно, CNN вимагають менших обчислювальних витрат і набагато швидше навчаються за рахунок паралельності.

Згорткові мережі складаються з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари CNN зазвичай складаються зі згорткових, агрегувальних і повзнозв'язних шарів, а також шарів нормалізації. Архітектура згорткової нейронної мережі зображена на рисунку 3.3.

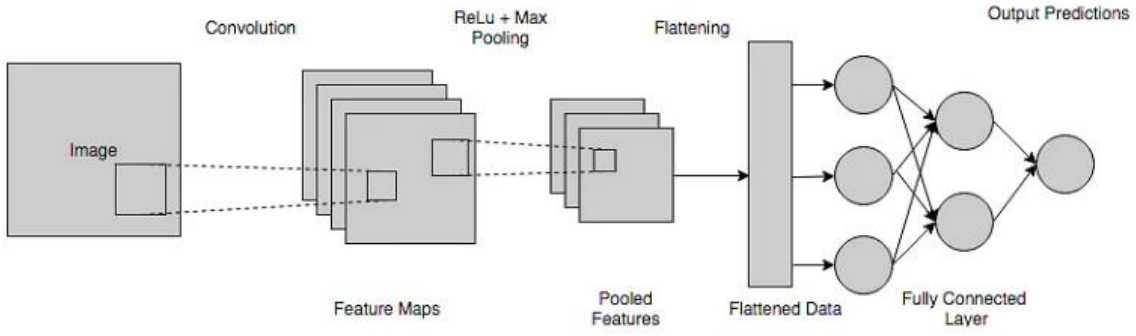


Рисунок 3.3 – Архітектура згорткової нейронної

Шар згортки (convolutional layer) – це основний блок згорткової мережі. Згорткові шари включають в себе для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар по фрагментам. Приклад виконання операції згортки подано на рисунку 3.4.

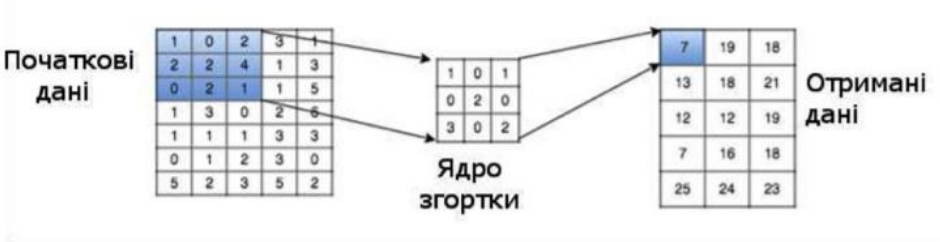


Рисунок 3.4 – Виконання операції згортки

Агрегувальні шари (pooling layer) є нелінійним ущільненням карти ознак. Для цього найбільш часто уживаною є функція максимуму. Операція пулінгу дозволяє суттєво зменшити просторовий обсяг зображення. Пулінг інтерпретується так: якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібне, і воно ущільнюється до менш детального. До того ж, фільтрація вже непотрібних деталей допомагає уникнути перенавчання.

У задачах з даними ЕКГ використовуються як 1D, так і 2D згорткові нейронні мережі. Одновимірні згорткові мережі працюють з часовими вимірами ЕКГ, тоді як двовимірні працюють з перетвореними двовимірними даними, наприклад, з 2D спектрограмами [7] або з одноканальними зображеннями [8].

3.2 Реалізація нейронних мереж для класифікації ЕКГ

Після детального аналізу існуючих рішень, наукових статей та власних спроб класифікації електрокардіограм було виявлено, що згорткові нейромережі показують одні з найкращих результатів. Однак основним недоліком двовимірних згорткових мереж є перетворення вхідних даних у двовимірні зображення. Отримані 2D зображення більш як на 75% складаються з білого кольору, тобто містять нулі, але беруть участь у обробці, чим збільшують кількість операцій із плаваючою комою. Такі мережі можуть бути покращені за рахунок використання 1D згорток.

Саме тому для побудови моделі була використана згорткова 1D нейронна мережа, тобто застосовувалась одновимірна згортка. Для цього з датасету було взято ~ 91% ЕКГ для тренування та ~ 9 % як тестову вибірку. Найпершим побудованим рішенням класифікації була розробка 2 нейронних мереж, перша призначена для виявлення зашумлених ЕКГ (клас ~ в датасеті), друга - для класифікації на 3 класи: нормальний ритм (N), миготливу аритмію

(AF) та інший вид ритму (O). Друге побудоване рішення базувалось на використанні ще 3 бінарних нейронних мереж для виявлення нормального ритму, аритмії та іншого виду ритму. Перевагою другого підходу є можливість розробки більш складного подальшого аналізу отриманих ймовірностей нейронних мереж для вибору результуючого класу.

3.2.1 Підготовка даних

При навчанні моделей великою проблемою стала мала кількість даних в вибірці та проблема незбалансованості класів. Остання проблема є типовою для електрокардіографії та викликає труднощі, пов'язані з навчанням нейронних мереж. Одним із способів подолання дисбалансу є класичні методи збільшення даних, такі як зсуви, повороти зображення та обертання, які стали важливим кроком у завданнях комп'ютерного зору. Проте такі перетворення можуть порушити залежність між деякими ознаками у сигналах, що негативно позначиться на якості цих даних, тому їх потрібно виконувати з обережністю.

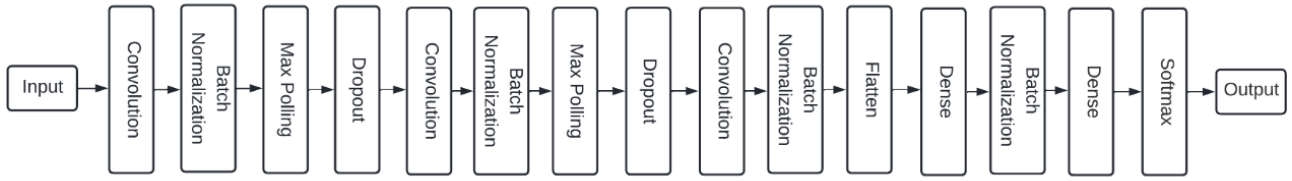
Для вирішення цих проблем було прийнято рішення збільшити кількість даних в деяких класах. Так для нейронної мережі, призначеної для виявлення зашумлених ЕКГ, дані з датасету нарізались по 900 значень. При цьому кількість даних в кожному класі була збільшена так, щоб збалансувати вибірку, а саме співвідношення класів стало 1:1:1:1. Для цього у випадкової ЕКГ брався її випадковий відрізок, який додавався у вибірку.

В свою чергу, для навчання мереж для бінарних класифікацій: “N” - “NotN”, “A” - “NotA”, “O” - “NotO”, а також мережі для класифікації на 3 класи (NAO): “N”, “AF”, “O”, ЕКГ розділялись на частини по кілька RR циклів з різними зсувами. Так для моделі на 3 класи бралось по 7 RR циклів, та для бінарних моделей було взято 5 RR циклів. Проблема такого підходу в тому, що довжина RR циклу не є фіксованою, в той час як нейронна мережа потребує

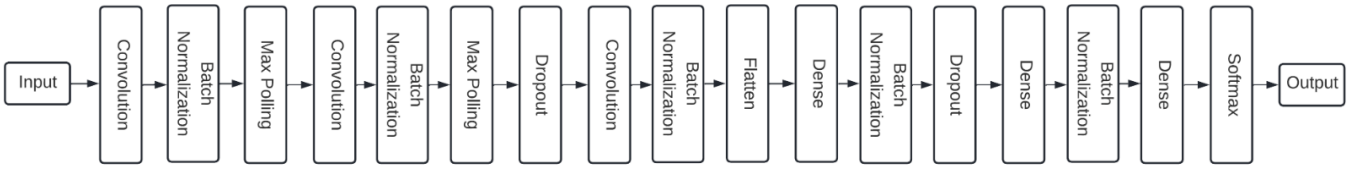
сталого розміру вхідних даних. Саме тому була виконана передискретизація отриманих довжин до сталих 1575 та 1100 значень відповідно.

3.2.2 Архітектура нейронних мереж

Розроблені нейронні мережі мають стандартну для згорткових мереж архітектуру, а саме складаються з чергування послідовних згорткових (Convolution) та агрегувальних (Max Pooling) шарів, а також шарів нормалізації (Batch Normalization). Також в мережах містяться кілька повнозв'язних (Dense) шарів в кінці та використовується dropout метод для зменшення перенавчання. Архітектура мереж зображена на діаграмах 3.5 та 3.6. Більш детальні діаграми можна побачити в додатку А.



Діаграма 3.5 – Архітектура мережі для класифікації на зашумлені та чисті ЕКГ



Діаграма 3.6 – Архітектура мережі для класифікації на класи "N", "A" та "O"

При навчанні моделей був використаний оптимізатор Адам, функція активації relu та softmax для вихідного шару. В якості функції помилки використовувалась категоріальна крос ентропія (categorical cross-entropy). Також для всіх згорткових шарів використовувався L2 регуляризатор, швидкість навчання була встановлена на рівні 0.001 з використанням експоненційного згасання (Exponential Decay).

3.3 Навчання моделей та результати

Найпершим етапом класифікації було визначення зашумлених кардіограм. Саме для цієї класифікації брались чисті дані без використання фільтрів, щоб не спотворювати результат. Отримана точність нейронної мережі становила 88%. ЕКГ визначені як зашумлені не брали більше участь в подальшій класифікації. Наступним етапом була класифікація на 3 класи за допомогою NAO мережі. Тут отримана точність становила 81% на тренувальному та 78% на тестовому наборах. Графіки навчання цих двох мереж зображено на рисунку 3.7, 3.8.

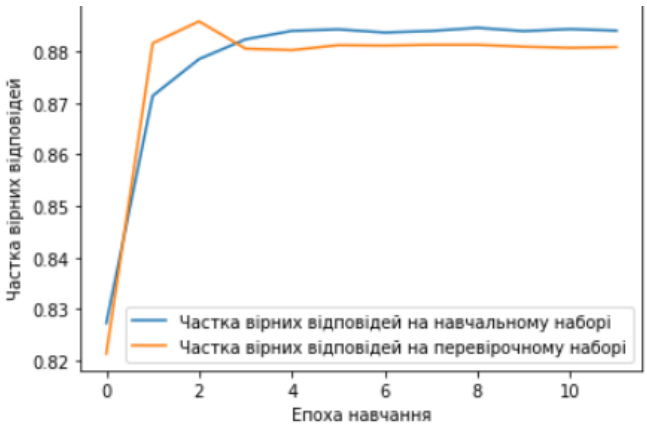


Рисунок 3.7– Графік точності “Noisy”-“NotNoisy” мережі протягом навчання

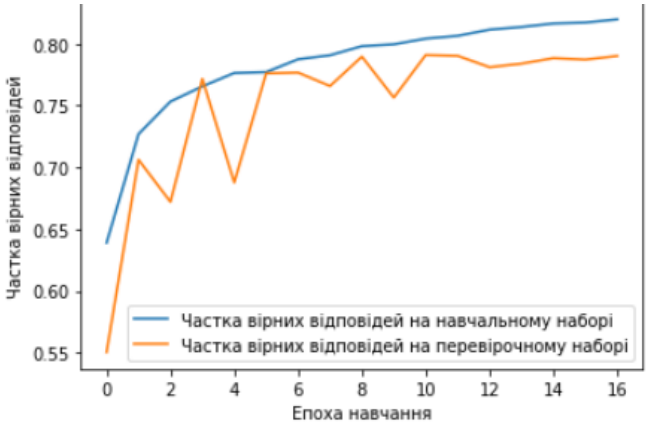


Рисунок 3.8– Графік точності NAO мережі протягом навчання

Так як в процесі класифікації брали участь нарізані сегменти кардіограми, результати по кожній частині поєднувались для всієї ЕКГ і лише потім обирався результуючий клас. В даній дипломній роботі було використано наступний підхід для вибору результуючого класу:

$$P_j = \prod_{i=1}^{n} p_{ij}$$

де n – кількість сегментів в класифікованій ЕКГ, j – клас, $0 \leq p_{ij} \leq 1$ – отримана ймовірність j – го класу для і-го сегменту. Для уникнення отримання замалих чисел була формула була модифікована наступним чином:

$$P_j = \sum_{i=1}^{n}(-\ln p_{ij}).$$

Остаточний клас для ЕКГ обирався такий, що має мінімальне значення P_j. В результаті отримана метрика $\boldsymbol{F_1}$ становила 0.8316. Результати класифікації відображені в таблиці 4.1.

Таблиця 4.1 – Результати класифікації нейронною мережею

F_{1n}	F_{1a}	F_{1o}	F_{1p}	F_1	F_1 valid.
0.8972	0.8502	0.7474	0,6167	**0.83163**	**0.8169**

Наступним побудованим рішенням було використання ще 3 бінарних нейронних мереж: “N” - “NotN”, “A” - “NotA”, “O” - “NotO”. Отримані точності для них становили 84.5%, 91.3% та 78% відповідно. Графіки навчання цих мереж зображено на рисунку 3.9 - 3.11.

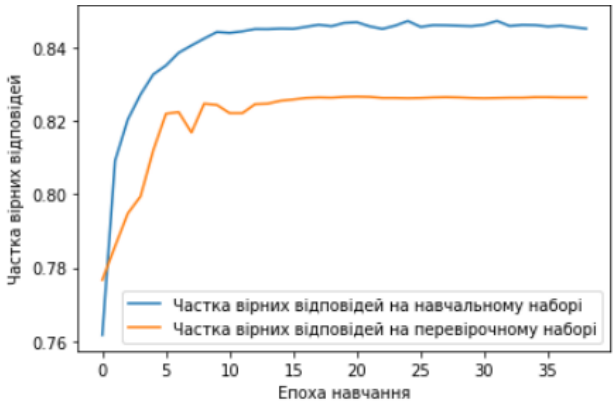


Рисунок 3.9– Графік точності “N”-“NotN” мережі протягом навчання

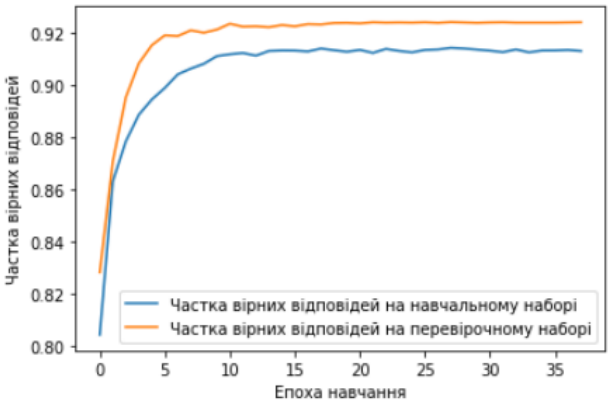


Рисунок 3.10– Графік точності “A”-“NotA” мережі протягом навчання

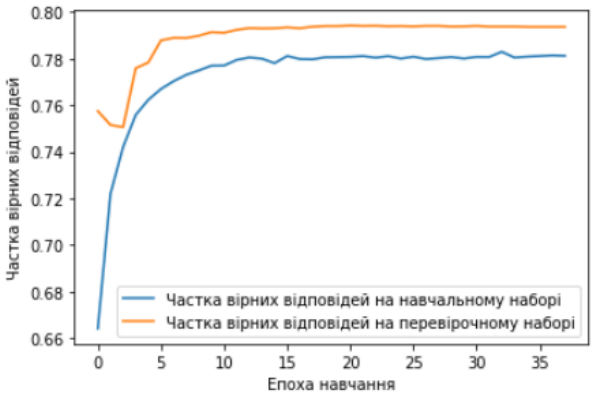


Рисунок 3.11– Графік точності “O”-“NotO”
мережі протягом навчання

На виході з нейронних мереж отримувались ймовірності $\boldsymbol{b_N, b_A, b_O}$ для бінарних класифікаторів і $\boldsymbol{t_N, t_A, t_O}$ для тернарного класифікатора. Щоб поєднати результати різних нейронних мереж були введені «зважені ймовірності» наступним чином $\boldsymbol{w_N = b_N\sqrt{t_N}}$, $\boldsymbol{w_A = b_A\sqrt{t_A}}$, $\boldsymbol{w_O = b_O\sqrt{t_O}}$. Остаточний клас вибирався на основі найбільшої «зваженої ймовірності». Отримана метрика $\boldsymbol{F_1}$в цьому випадку становила 0.8330. Результати класифікації відображені в таблиці 4.2.

Таблиця 4.2 – Результати класифікації, використовуючи зважені ймовірності

F_{1n}	F_{1a}	F_{1o}	F_{1p}	F_1	F_1 valid.
0.8975	0.8531	0.7485	0,6167	**0.8330**	**0.8243**

РОЗДІЛ 4 ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ ЕКГ

В попередньому розділі було описано як виконувалась класифікація ЕКГ, базуючись на результатах 3 бінарних нейронних мереж та 1 тернарної мережі, використовуючи «зважені ймовірності». Проте існує ще багато методів машинного навчання, що можуть бути використані для аналізу отриманих з нейронних мереж ймовірностей. В рамках цієї дипломної роботи були розроблені та протестовані класифікатори на основі випадкових дерев, методу опорних векторів, логістичної регресії та наївного баєсівого класифікатора. Для всіх класифікаторів була використана бібліотека SKLearn. Кожен метод навчався на кортежах з 6, 9 та 21 параметром. Кортежі параметрів складалися наступним чином:

$$(\boldsymbol{b_N, b_A, b_O, t_N, t_A, t_O})$$

$$(\boldsymbol{b_N, b_A, b_O, t_N, t_A, t_O, w_N, w_A, w_O})$$

$$(\boldsymbol{b_N, b_A, b_O, t_N, t_A, t_O, w_N, w_A, w_O, b_N^2, b_A^2, b_O^2, t_N^2, t_A^2, t_O^2,}$$
$$\boldsymbol{\sqrt{b_N b_A}, \sqrt{b_N b_O}, \sqrt{b_A b_O}, \sqrt{t_N t_A}, \sqrt{t_N t_O}, \sqrt{t_A t_O}})$$

де $\boldsymbol{b_N, b_A, b_O}$ - ймовірності бінарних класифікаторів, $\boldsymbol{t_N, t_A, t_O}$- ймовірності тернарного класифікатора, $\boldsymbol{w_N = b_N\sqrt{t_N}, w_A = b_A\sqrt{t_A}, w_O = b_O\sqrt{t_O}}$ - зважені ймовірності.

4.1 Наївний баєсів класифікатор

Наївний баєсів класифікатор — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні незалежності змінних. Цей класифікатор вважається одним із найпростіших з алгоритмів класифікації. Тим не менш, дуже часто він працює не гірше та значно швидше за складніші алгоритми.

Теорема Байеса — це математичне рівняння, яке використовується для обчислення умовної ймовірності. Іншими словами, теорема використовується для обчислення ймовірності події на основі її зв'язку з іншою подією.

Теорема Баєса задається математично таким рівнянням:

$$\boldsymbol{P(A|B) = \frac{P(B|A)P(A)}{P(B)}},$$

де A та B є певними подіями, $P(A)$ та $P(B)$ є ймовірностями настання події A та B, $P(A|B)$ – ймовірність A за умови істинності B, $P(B|A)$ – ймовірність спостереження події B за умови істинності A.

Основним припущенням наївного баєсівого алгоритму є те, що кожна характеристика вносить незалежний та рівний внесок у кінцевий результат. Однак таке допущення, як правило, некоректне в реальних ситуаціях, але часто добре працює на практиці. Тому алгоритм і називається наївним.

Отже, наївний Байєс є умовною ймовірнісною моделлю: нехай заданий екземпляр, який потрібно класифікувати, представлений вектором $\boldsymbol{x = (x_1, \dots, x_n)}$, що представляє деякі **n** ознак (незалежних змінних), він призначає цьому екземпляру ймовірності $P(C_k|x1, \dots xn)$ для кожного з **K** можливих класів $\boldsymbol{C_k}$. Тоді застосувавши теорему Баєса маємо:

$$\boldsymbol{P(C_k|x_1, \dots, x_n) = \frac{P(C_k)P(x_1, \dots, x_n|C_k)}{P(x_1, \dots, x_n)}}$$

На практиці цікавий лише чисельник цього дробу, еквівалентний спільній ймовірнісній моделі, оскільки знаменник не залежить від $\boldsymbol{C_k}$, а значення ознак $\boldsymbol{x_i}$ дано, а отже знаменник - константа. Тепер вступають в гру «наївні» умовні припущення незалежності, що означає:

$$\boldsymbol{P\left(x_i|x_j, C_k\right) = P(x_i|C_k)}.$$

Таким чином, модель може бути виражена як:

$$\boldsymbol{P(C_k|x_1,\dots,x_n) = \frac{1}{Z}P(C_k)\prod_{i=1}^{n}P(x_i|C_k),}$$

де $\boldsymbol{Z = P(x_1,\dots,x_n)}$.

Наївний баєсів класифікатор об'єднує модель з правилом рішення, що полягає в виборі найбільш ймовірної гіпотези. Відповідний класифікатор визначений таким чином:

$$\boldsymbol{\hat{y} = arg\max_k P(C_k)\prod_{i=1}^{n}P(x_i|C_k)}$$

У випадку з ймовірностями витягнутими з нейронних мереж ми маємо справу з безперервними даними. Тому для класифікації було зроблено типове припущення, що ці значення мають нормальний розподіл, тоді умовна ймовірність визначалась наступним чином:

$$\boldsymbol{P(x) = \frac{1}{\sqrt{1\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}},}$$

де μ та σ^2 будуть відповідно визначатись як математичне очікування та дисперсія.

Результати класифікації на кортежах з 6, 9, 21 параметром подано в таблиці 4.3.

Таблиця 4.3 – Результати класифікації за допомогою наївного баєсового класифікатора

№ params	F_{1n}	F_{1a}	F_{1o}	F_1	F_1 valid.
6	0.8937	0.8589	0.7352	**0.8293**	**0.8177**
9	0.8963	0.8569	0.7438	**0.8323**	**0.8202**
21	0.8959	0.855	0.7363	**0.8290**	**0.8193**

4.2 Логістична регресія

Логістична регресія — це різновид множинної регресії, загальне призначення якої полягає в аналізі зв'язку між декількома незалежними змінними (так званими регресорами або предикторами) і залежною змінною. Основна відмінність лінійної регресії від логістичної регресії полягає в тому, що діапазон логістичної регресії обмежений між 0 і 1. Крім того, на відміну від лінійної регресії, логістична регресія не вимагає лінійного відношення між вхідними та вихідними змінними.

Логістична регресія застосовується для прогнозування ймовірності виникнення деякої події за значенням множини ознак. Для бінарної класифікації вводиться залежна змінна $\boldsymbol{y}$, що приймає значення 0 і 1 і безліч незалежних змінних $\boldsymbol{x_1, \dots, x_n}$. В цьому випадку логістична регресія використовує логістичну функцію (сігмоїду) для моделювання бінарної вихідної змінної:

$$\boldsymbol{f(X) = \frac{1}{1 + e^{-X}} = \frac{1}{1 + e^{-\sum_{i=0}^{n} w_i x_i}}}$$

$\boldsymbol{f(X)}$ в цьому випадку показує ймовірність того, що $\boldsymbol{y}$ належить класу $\boldsymbol{1}$.

Задача навчання лінійного класифікатора полягає у тому, щоб для вибірки $\mathbf{X^m}$ налаштувати вектор ваг $\mathbf{w}$. У логістичній регресії при цьому вирішується задача мінімізації емпіричного ризику з функцією втрат спеціального виду:

$$\boldsymbol{Cost = -y * ln\left(f\left(x\right)\right) - (1 - y) * ln(\,1 - f(x))}$$

Тоді загальні втрати на всіх об’єктах вибірки дорівнюють:

$$\boldsymbol{J(W) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} * ln\left(f\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) * ln(\,1 - f\left(x^{(i)}\right))}$$

Підбір оптимальних ваг може виконуватись за допомогою стохастичного градієнтного спуску наступним чином:

$$\boldsymbol{w_{ij}} = \boldsymbol{w_{ij}} - \boldsymbol{\alpha}\frac{\boldsymbol{\partial J}}{\boldsymbol{\partial w_{ij}}}$$

де $\boldsymbol{J}$– функція помилки, $\boldsymbol{\alpha}$ – крок методу.

Аналогічно для задач класифікації на декілька класів використовується функція Softmax. Регресія Softmax – це узагальнення логістичної регресії, яке можна використовувати для класифікації на K класів. Функція Softmax використовується для відображення K-мірного вектору довільних дійсних значень в інший K-мірний вектор дійсних значень, де кожен елемент вектору знаходиться в інтервалі (0, 1). Таким чином присвоюється ймовірність кожному класу в мультикласовій задачі. Ці ймовірності мають становити в сумі 1.

$$\boldsymbol{Softmax(X_i)} = \frac{\boldsymbol{e^{X_i}}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{K}} \boldsymbol{e^{X_j}}}$$

Узагальнюючи, логістичну регресію можна представити у вигляді одношарової нейронної мережі з сигмоїдальною функцією активації, вагами якої є коефіцієнти логістичної регресії. Представлення логістичної регресії у вигляді нейронної мережі зображено на рисунку 4.1.

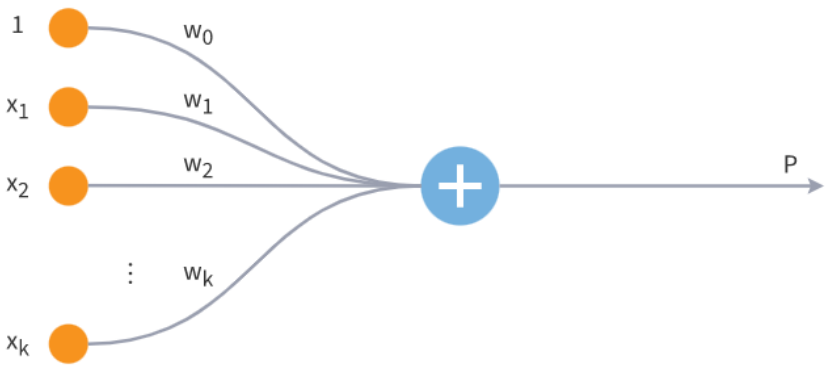


Рисунок 4.1– Представлення логістичної регресії у вигляді нейронної мережі

Результати класифікації за допомогою логістичної регресії на кортежах, використовуючи 6, 9 та 21 параметр подано в таблиці 4.4.

Таблиця 4.4 – Результати класифікації з використанням логістичної регресії

№ params	F_{1n}	F_{1a}	F_{1o}	F_1	F_1 valid.
6	0.9031	0.8677	0.7672	**0.846**	**0.8194**
9	0.9028	0.8709	0.7675	**0.8471**	**0.819**
21	0.9043	0.8715	0.7733	**0.8497**	**0.8195**

4.3 Метод опорних векторів

Метод опорних векторів (Support Vector Machine – SVM) – це дуже потужний та універсальний метод машинного навчання, здатний виконувати лінійну чи нелінійну класифікацію, регресію і навіть виявлення викидів. SVM особливо добре підходить для класифікації складних, але невеликих чи середніх наборів даних.

Основним завданням алгоритму є знайти найбільш правильну гіперплощину, що розділяє дані на два класи $Y \epsilon \{-1; 1\}$. Іншими словами, мета SVM як класифікатора - знайти рівняння роздільної гіперплощини в просторі $\boldsymbol{R^n}$

$$\boldsymbol{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_0 = 0,}$$

яка розділила б два класи оптимальним чином. Тоді вибір класу для деякого об’єкта $\boldsymbol{x}$ буде відбуватись наступним чином:

$$\boldsymbol{F(x) = sign(w^T x + w_0)}$$

Після налаштування ваг алгоритму в процесі навчання всі об'єкти, що потрапляють по один бік від побудованої гіперплощини, будуть класифікуватись як перший клас, а об'єкти, що потрапляють по інший бік — другий клас.

Розділяючу гіперплощину можна побудувати різними способами, але в SVM ваги налаштовуються таким чином, щоб об'єкти класів лежали якнайдалі від роздільної гіперплощини. Іншими словами, алгоритм максимізує зазор (margin) між гіперплощиною та об'єктами класів, які розташовані найближче до неї. Такі об'єкти називають опорними векторами, звідки й назва алгоритму. Опорні вектори в цьому випадку будуть відповідати наступній рівності:

$$\boldsymbol{y_i(w^T x_i + w_0) = 1, \qquad y_i \epsilon \{-1; 1\}}$$

В свою чергу відстань між двома опорними векторами буде дорівнювати $\frac{2}{\|w\|}$.

Демонстрацію побудови опорних векторів зображено на рисунку 4.2.

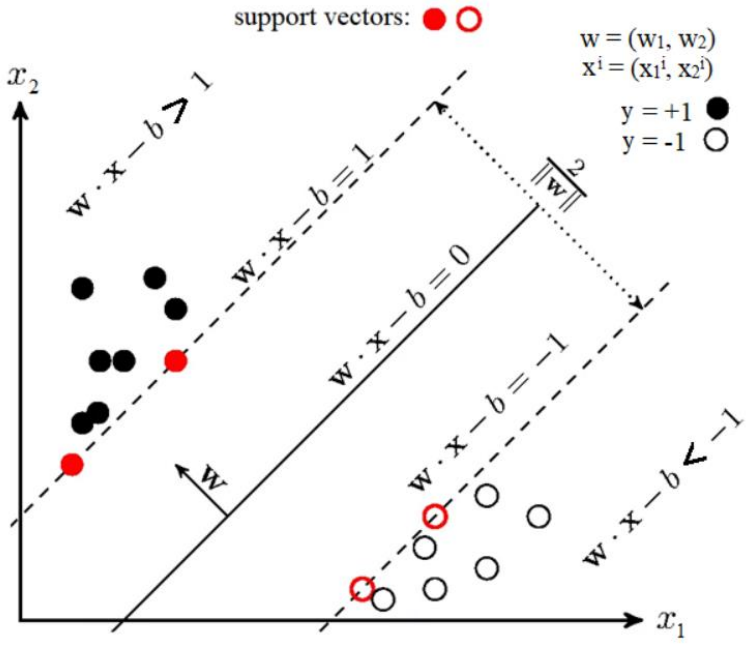


Рисунок 4.2 – Демонстрація побудови опорних векторів

Таким чином основне завдання SVM алгоритму полягає у знаходженні вектору $\boldsymbol{w}$ так, щоб гіперплощина розділяла вибірку і ширина зазору між класами була максимальною. Чим більше значення ширини зазору тим класифікатор впевненіший у відповіді. Тому математична постановка задачі виглядає наступним чином:

$$\begin{cases} \|\boldsymbol{w}\| \rightarrow \boldsymbol{min} \\ \boldsymbol{y_i}(\boldsymbol{w^T x_i} + \boldsymbol{w_0}) \geq \boldsymbol{1} \end{cases}$$

Проте друга умова рідко коли виконується та лінійно розділити вибірку на 2 класи часто не вдається. Саме тому вводиться послаблення умови так, що об'єктам з одного класу дозволено попадати в область іншого класу:

$$\begin{cases} \|\boldsymbol{w}\| \rightarrow \boldsymbol{min} \\ \boldsymbol{y_i}(\boldsymbol{w^T x_i} + \boldsymbol{w_0}) \geq \boldsymbol{1} - \boldsymbol{\varepsilon_i}, \forall \boldsymbol{i} \in \overline{\boldsymbol{1,n}} \\ \boldsymbol{\varepsilon_i} \geq \boldsymbol{0}, \boldsymbol{i} = \overline{\boldsymbol{1,n}} \end{cases}$$

Але такі хибні попадання треба штрафувати, щоб похибка не була занадто великою. У результаті отримаємо таку задачу:

$$\begin{cases} \frac{1}{2}\|\boldsymbol{w}\|^2 + \mathrm{C}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{\varepsilon_i} \rightarrow \boldsymbol{min} \\ \boldsymbol{y_i}(\boldsymbol{w^T x_i} + \boldsymbol{w_0}) \geq \boldsymbol{1} - \boldsymbol{\varepsilon_i}, \forall \boldsymbol{i} \in \overline{\boldsymbol{1,n}} \\ \boldsymbol{\varepsilon_i} \geq \boldsymbol{0}, \boldsymbol{i} = \overline{\boldsymbol{1,n}} \end{cases},$$

де параметр C відповідає за ціну помилки класифікації.

Однак в такого підходу є недолік, він не працює в випадку, коли дані в вибірці по своїй природі не можуть бути розділеними гіперплощиною. В цьому випадку застосовується так званий ядровий трюк (Kernel Trick). Таким чином відбувається перетворення нелінійного простору нижньої розмірності в простір вищої розмірності, так, що ми змогли отримати лінійну класифікацію. Візуалізація ідеї такого трюку зображена на рисунку 4.3.

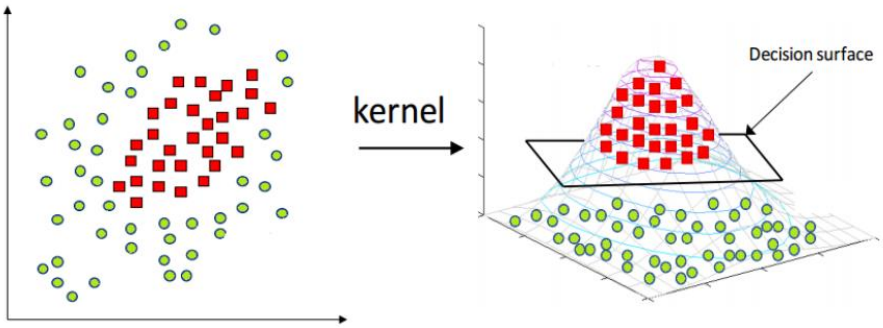


Рисунок 4.3 – Застосування ядра

Найбільш часто використовуються такі ядра: лінійне, поліноміальне, сигмоїдне та гаусове (з радіальною базовою функцією - RBF). Це перелік «стандартних» ядер, які при ближчому розгляді приводять до вже відомих алгоритмів: поліноміальних розділяючих поверхонь, двошарових нейронних мереж, RBF-мережам та іншим. Таким чином, ядра претендують на роль універсальної мови для опису широкого класу алгоритмів машинного навчання. Тут насправді спостерігається парадоксальна ситуація. З одного боку, ядра - одне з найкрасивіших винаходів у машинному навчанні. З іншого боку, досі не знайдено ефективного загального підходу до їх підбору в конкретних задачах.

В даній дипломній роботі для класифікації за допомогою SVM було взято RBF ядро:

$$\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{e}^{-\frac{\|x - x'\|^2}{2\sigma^2}}$$

Результати класифікації методом опорних векторів на кортежах, використовуючи 6, 9 та 21 параметр подано в таблиці 4.5.

Таблиця 4.5 – Результати класифікації методом опорних векторів

№ params	F_{1n}	F_{1a}	F_{1o}	F_1	F_1 valid.
6	0.9049	0.8736	0.7682	**0.8489**	**0.8194**
9	0.9023	0.8705	0.7619	**0.8449**	**0.8263**
21	0.9022	0.8712	0.7632	**0.8455**	**0.8278**

4.4 Випадковий ліс

Випадковий ліс - алгоритм машинного навчання, що працює за рахунок побудови ансамблю дерев прийняття рішень. У порівнянні з іншими методами машинного навчання теоретична частина алгоритму Random Forest проста.

Кожне дерево у випадковому лісі повертає прогноз класу, і клас із найбільшою кількістю голосів стає прогнозом лісу.

Алгоритм побудови випадкового лісу складається з чотирьох етапів:

1. Створення випадкових вибірок із заданого набору даних.
2. Для кожної вибірки будується дерево рішень і отримується результат прогнозу, використовуючи це дерево.
3. Проводиться голосування за кожен отриманий прогноз.
4. Обирається прогноз з найбільшою кількістю голосів як кінцевий результат.

Ілюстрація цього алгоритму зображено на рисунку 4.4.

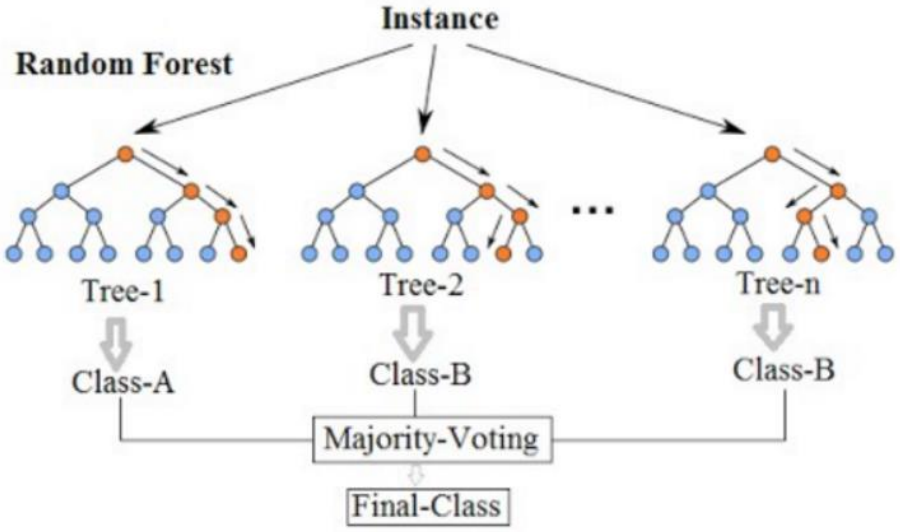


Рисунок 4.4 – Ілюстрація роботи випадкового лісу

Загальна схема побудови дерева прийняття рішень виглядає наступним чином:

• обираємо чергову ознаку **Q**, по якому буде відбуватись розділення, та поміщаємо її в корінь;

• для всіх її значень ***i***:

- залишаємо з тренувального набору лише ті, які мають значення атрибута **Q** рівним ***i;***
- рекурсивно будуємо дерево для цього нащадка.

В свою чергу вибір атрибута **Q**, по якому буде відбуватись розбиття, повинен здійснюватися за певним правилом. Критеріїв існує багато, але найбільшою популярністю користуються теоретико-інформаційний та статистичний.

Теоретико-інформаційний критерій, як випливає з назви, заснований на поняттях теорії інформації, а саме інформаційної ентропії:

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \ln\left(\frac{N_i}{N}\right),$$

де n - число класів у вихідній підмножині, N_i - число прикладів і-го класу, N - загальна кількість прикладів в підмножині.

Таким чином, ентропія може розглядатися як міра неоднорідності підмножини за представленими у ньому класами. Коли класи представлені у рівному співвідношенні, невизначеність класифікації найбільша та ентропія максимальна. А отже найкращим атрибутом розбиття **Q**j буде той, який забезпечить мінімальне значення ентропії результуючої підмножини.

В основі статистичного підходу лежить використання індексу Джіні. Статистичний сенс цього показника в тому, що він показує, наскільки часто випадково обраний приклад навчальної множини буде розпізнаний неправильно, за умови, що цільові значення в цій множині були взяті з певного статистичного розподілу. Таким чином, індекс Джині фактично показує відстань між двома розподілами — розподілом цільових значень, і розподілом передбачень моделі. Очевидно, чим менше дана відстань, тим краще працює модель. Індекс Джіні може бути розрахований за такою формулою:

$$Gini(Q) = 1 - \sum_{i=1}^{n} {p_i}^2$$

де Q – результуюча множина, n – число класів, pi – ймовірність i-го класу.

Теоретично, алгоритм навчання дерева рішень буде працювати доти, доки в результаті не буде отримано абсолютно «чистих» підмножин, у кожному з яких будуть приклади одного класу. Проте очевидним недоліком такої зупинки є перенавчання, саме тому вирішенням проблеми є примусова зупинка побудови дерева. Основними розробленими підходами такої зупинки є рання зупинка (алгоритм буде зупинено, як тільки буде досягнуто заданого значення деякого критерію, наприклад, відсоткової частки правильно розпізнаних прикладів), обмеження глибини (завдання максимальної кількості розбиття у гілках) та задання мінімально допустимого числа прикладів у вузлі.

Причина, через яку модель випадкового лісу працює так добре, полягає в тому, що велика кількість відносно некорельованих дерев, що працюють разом, перевершуватиме будь-яку з їх окремих складових. Ключовим фактором є слабка кореляція між деревами. Причиною такого ефекту є те, що дерева захищають одне одного від своїх індивідуальних помилок, принаймні доти, доки вони не будуть постійно помилятися в тому самому випадку. Для гарантії відсутності кореляції випадковий ліс два методи: беггінг та випадковість ознак.

Ідея беггінгу полягає в навчанні одного алгоритму багато разів на випадкових вибірках вхідних даних. Дані в випадкових вибірках можуть повторюватися. Тобто з набору 1-2-3 ми можемо робити вибірки 2-2-3, 1-2-2, 3-1-2 і так поки не набридне. Дерева рішень дуже чутливі до даних, на яких навчаються, невеликі зміни в наборі можуть призвести до деревоподібних структур, що значно відрізняються. Випадковий ліс використовує цю

перевагу, дозволяючи кожному окремому дереву довільно обирати дані, що призводить до різних дерев.

Випадковість ознаки полягає в тому, що кожне дерево в випадковому лісі може обирати лише з випадкової підмножини об’єктів. У звичайному дереві рішень, коли потрібно розділити вузол, ми розглядаємо кожну можливу ознаку і вибираємо ту, яка сильніше ділить значення у вузлах. Але саме застосування принципу випадковості ознак призводить до ще більшої варіації між деревами в моделі та зрештою до більш слабкої кореляції між деревами та більшої різноманітності.

Результати класифікації за допомогою випадкового лісу з використанням 6, 9 та 21 параметру подано в таблиці 4.6.

Таблиця 4.6 – Результати класифікації з використанням випадкового лісу

№ params	F_{1n}	F_{1a}	F_{1o}	F_1	F_1 valid.
6	0.9038	0.8728	0.7675	**0.848**	**0.8246**
9	0.905	0.8686	0.7718	**0.8485**	**0.8225**
21	0.905	0.869	0.774	**0.8493**	**0.8173**

4.5 Результати

Загальні результати класифікації розглянутими підходами подані в таблиці 4.7.

Таблиця 4.7 – Результати класифікації

Назва методу	№ params	$\boldsymbol{F_{1n}}$	$\boldsymbol{F_{1a}}$	$\boldsymbol{F_{1o}}$	$\boldsymbol{F_1}$	$\boldsymbol{F_1}$ valid.
NAO	-	0.8972	0.8502	0.7474	0.8316	0.8169
Max $\boldsymbol{w_x}$	6	0.8975	0.8531	0.7485	0.8330	0.8243
Bayes	6	0.8937	0.8589	0.7352	0.8293	0.8177
Bayes	9	0.8963	0.8569	0.7438	0.8323	0.8202
Bayes	21	0.8959	0.855	0.7363	0.8290	0.8193
Log. reg.	6	0.9031	0.8676	0.7672	0.8460	0.8194

Log. reg.	9	0.9028	0.8709	0.7674	0.8470	0.8219
Log. reg.	21	0.9042	0.8715	0.7733	**0.8497**	0.8195
Rand. forest	6	0.9037	0.8728	0.7674	0.8480	0.8246
Rand. forest	9	**0.9050**	0.8686	0.7718	0.8485	0.8225
Rand. forest	21	**0.9050**	0.8690	**0.7740**	0.8493	0.8173
SVM	6	0.9048	**0.8735**	0.7682	0.8489	0.8205
SVM	9	0.9023	0.8705	0.7619	0.8449	0.8262
SVM	21	0.9022	0.8712	0.7632	0.8455	**0.8278**

Як можна побачити з таблиці, використання бінарних нейронних мереж з подальшим аналізом отриманих ймовірностей дає кращі результати в порівнянні зі звичайним використанням тернарної мережі як для тренувального, так і для тестового набору. Максимальне значення в метриці **F1** показало використання логістичної регресії на кортежі з 21 параметром. Найкращих результатів вдалось досягнути у виявленні нормального синусового ритму та миготливої аритмії, 0.9050 (Rand. Forest 9) та 0.8735 (SVM 6) відповідно. Більш детальну таблицю результатів можна зайти в додатку Б.

В таблиці 4.8 представлено порівняння отриманих результатів з попередніми статтями.

Таблиця 4.8 – Порівняння запропонованої моделі з попередніми рішеннями.

Джерело	$\boldsymbol{F_{1n}}$	$\boldsymbol{F_{1a}}$	$\boldsymbol{F_{1o}}$	$\boldsymbol{F_{1p}}$	$\boldsymbol{F_1}$
[9]	0.931	0.870	0.839	-	0.880
[10]	0.9030	0.8547	0.7366	0.5622	0.83
[11]	0.9024	0.8156	0.7194	0.5707	0.81
[12]	0.8912	0.7673	0.6651	0.4368	0.77
(Log. reg. 21)	**0.9042**	**0.8715**	**0.7733**	**0.6106**	**0.8497**

ВИСНОВКИ

Метою даної дипломної роботи було дослідження існуючих підходів для якісного автоматизованого аналізу ЕКГ сигналів та огляд популярних методів машинного навчання, що можуть бути застосовані для класифікації серцевих захворювань.

В даній роботі було виконано ряд завдань:

- досліджено відомі алгоритми фільтрації дискретних сигналів; реалізовано рухоме середнє і фільтр Баттерворта для очищення ЕКГ від шуму;
- розроблено алгоритм локалізації R-піків;
- проведений аналіз та порівняння різних методів машинного навчання для класифікації кардіограм;
- розглянуто різні архітектури глибоких нейронних мереж та натреновано 5 згорткових мереж по визначених типах;
- розроблено класифікатори на основі методу опорних векторів, випадкових дерев та логістичної регресі як ансамблі нейромереж;
- розроблений власний підхід до класифікації ЕКГ сигналів, який показав конкурентні результати в порівнянні з існуючими рішеннями.

Розроблений метод класифікації ЕКГ-сигналів складається з чотирьох основних етапів:

1) виявлення та відкидання зашумлених кардіограм;

2) попередня обробка ЕКГ, що включає в себе застосування фільтрів, пошук R-піків, нарізання ЕКГ на менші частини та передискретизацію;

3) класифікація ЕКГ бінарними та тернарною нейронними мережами;

4) остаточний вибір класу, використовуючи відомі алгоритми машинного навчання, на базі отриманих ймовірностей.

Отриманий результат в досягнув позначки 0.8497 метриці $\boldsymbol{F_1}$. Розроблений підхід також має хороші перспективи в можливому подальшому його використанні на вбудованих системах для тривалих безперервних кардіологічних спостережень. Результати даної роботи відображені в публікації [13].

Планується покращення даного рішення, в планах розробка алгоритму локалізації P і T хвиль з їх подальшим аналізом. Також в планах розширення можливостей рішення шляхом навчання моделей на більших датасетах з можливістю класифікації більшої кількості захворювань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Goldberger A. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals / Amaral L., Glass L., Hausdorff J., Ivanov P. C., Mark R., Stanley H. E. // Circulation [Online]. 101 (23). – 2000. – pp. 215–220.
2. Clifford G. D. AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017 / Liu C., Moody B., Liwei H. L., Silva I., Li Q., Johnson A. E., Mark R. G. // Computing in Cardiology (CinC). – 2017. – pp. 1– 4.
3. Moody G. B. The impact of the MIT-BIH Arrhythmia Database / Mark R. G. // IEEE Eng in Med and Biol 20(3). – 2001. – pp. 45-50.
4. Wu L. ECG Enhancement and R-Peak Detection Based on Window Variability / Xie X., Wang Y. // Healthcare 9(2): 227. – 2021.
5. Singh S. Classification of ECG arrhythmia using recurrent neural networks / Pandey S. K., Pawar U., Janghel R. R. // Procedia Computer Science. – 2018. – pp. 1290-1297.
6. Saadatnejad S. Lstm-based ECG classification for continuous monitoring on personal wearable devices / Oveisi, M., Hashemi, M. // IEEE journal of biomedical and health informatics. – 2020. – pp. 515-523.
7. Zihlmann M. Convolutional recurrent neural networks for electrocardiogram classification / Perekrestenko D., Tschannen M. // Computing in Cardiology (CinC). – 2017. – pp. 1-4.
8. Jun T. J. ECG arrhythmia classification using a 2-D convolutional neural network // arXiv: Cornell University. – 2018.

9. Chen D. Electrocardiogram Classification and Visual Diagnosis of Atrial Fibrillation with DenseECG / Li D., Xu X. // arXiv: Cornell University. – 2021.
10. Teijeiro T. Arrhythmia classification from the abductive interpretation of short single-lead ecg records / García C. A., Castro D. // Computers in Cardiology, 44. – 2017. – pp. 1–4.
11. Goodfellow S. D. Classification of atrial fibrillation using multidisciplinary features and gradientboosting / Goodwin A., Greer R., Laussen P. C., Mazwi M., Eytan D. // Computing in Cardiology (CinC). –2017. – pp. 1–4.
12. Whitaker, B. AF Classification from ECG Recording UsingFeature Ensemble and Sparse Coding / Rizwan M., Aydemir B., Rehg J., Anderson D. // In Proceedings of the Computing in Cardiology Conference. – 2017.
13. Харченко В.В. Класифікація ЕКГ сигналів методами машинного навчання // Вісник Київського національного університету імені Тараса Шевченка. Серія: фіз.-мат. науки. – 2022. – вип. 2.

ДОДАТОК А

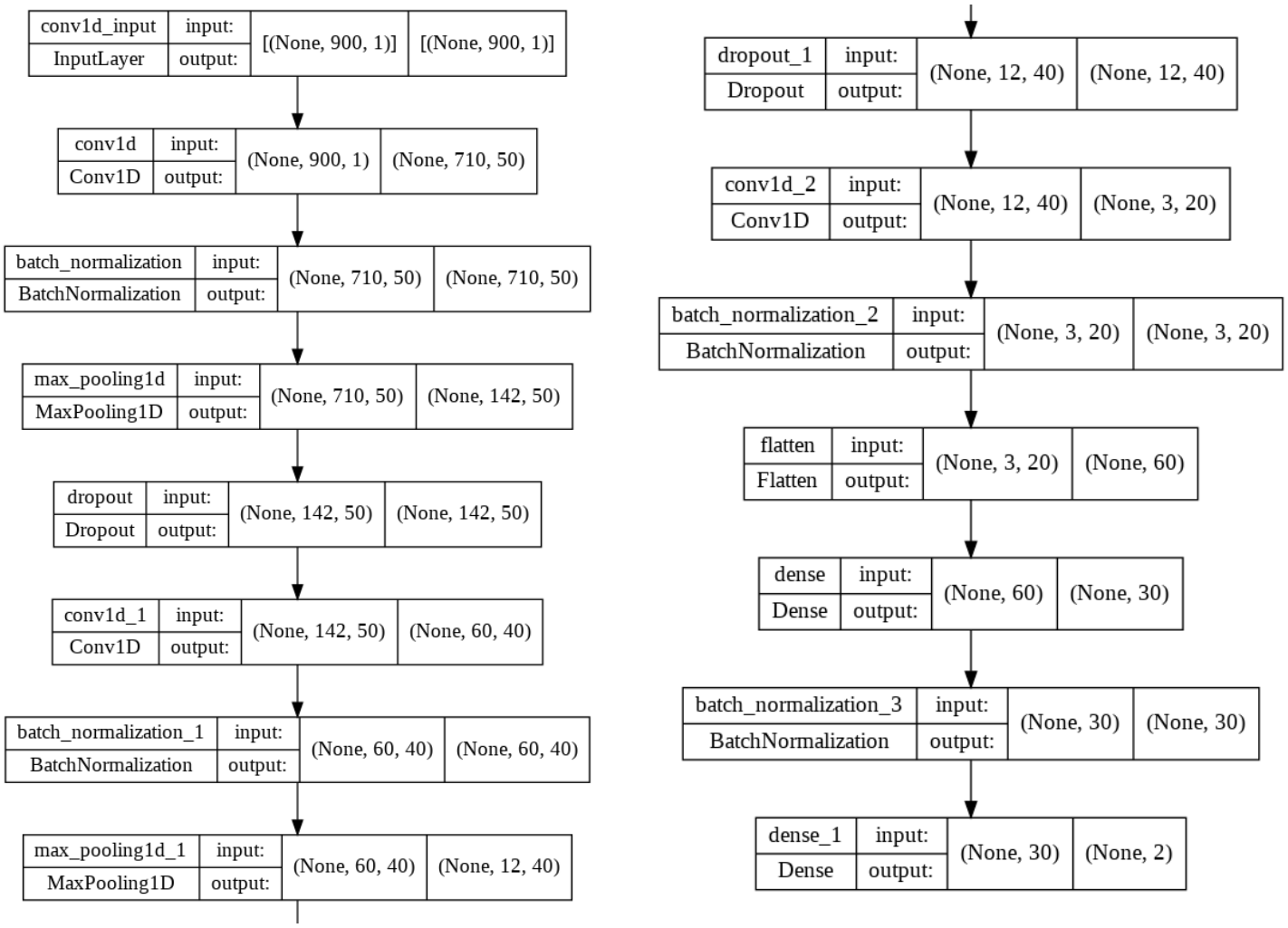


Рисунок 4– Структура нейронної мережі для класифікації на зашумлені та чисті ЕКГ

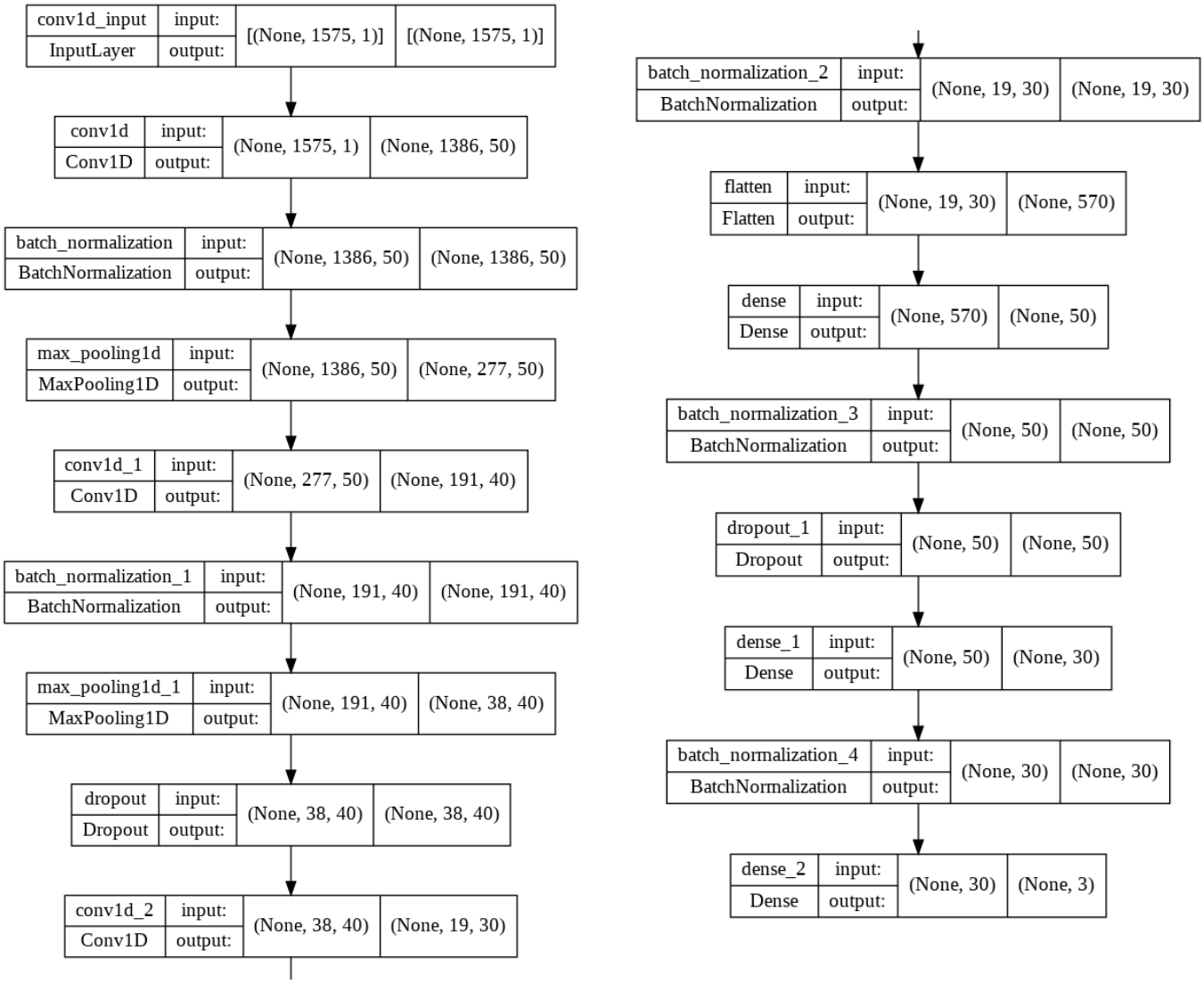


Рисунок 5 – Структура NAO мережі для тернарної класифікації

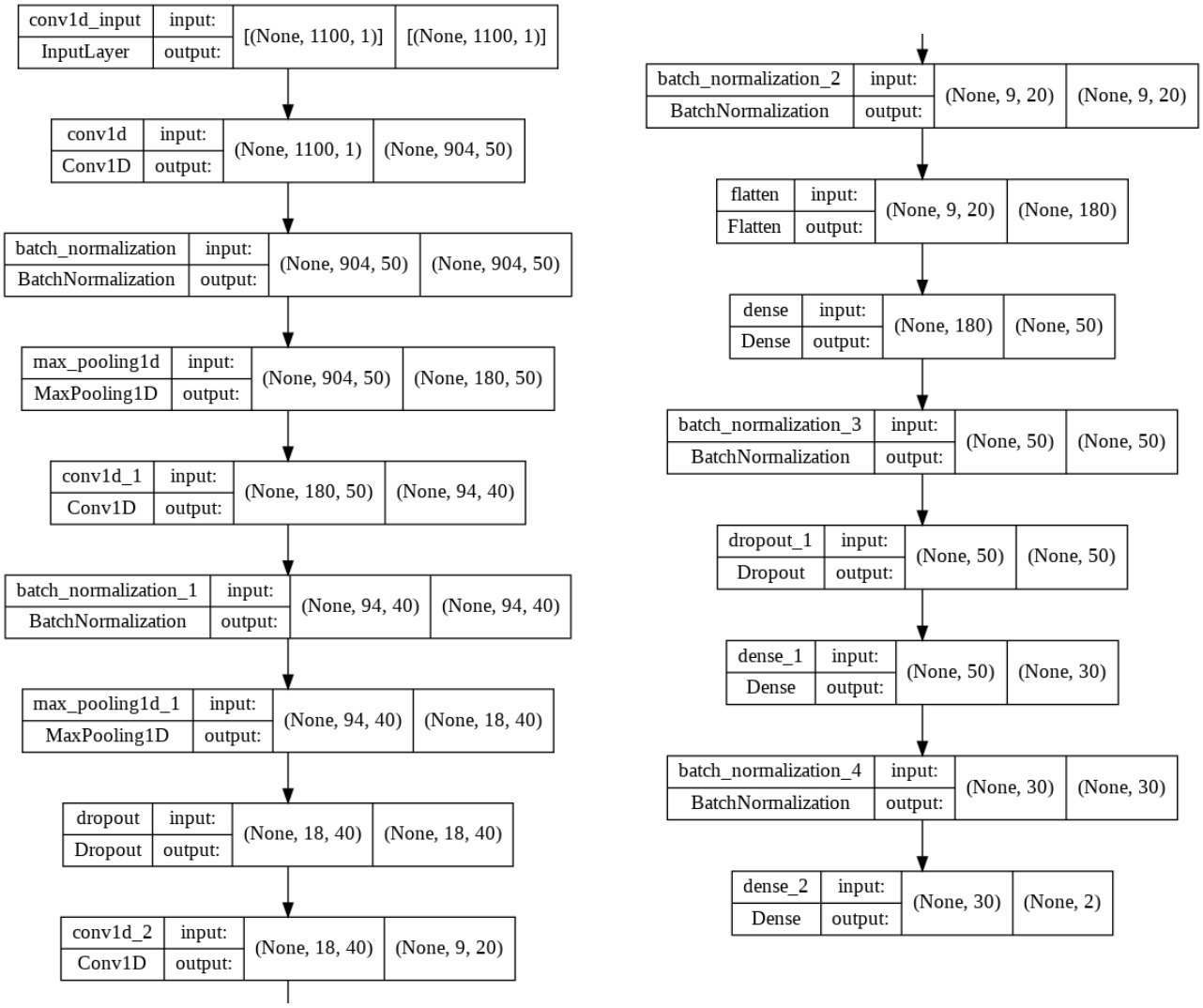


Рисунок 6 – Структура “N-notN”, “A-notA”, “O-notO” мереж для бінарних класифікацій

ДОДАТОК Б

Таблиця 5 - Загальні результати для всіх розглянутих класифікацій

Method	Full Dataset	Validation
NAO NN	Fn = 0.897216484705419 Fa = 0.8502122498483929 Fo = 0.747470303563572 **F1 = 0.831633012705795**	Fn = 0.8897338403041825 Fa = 0.8444444444444444 Fo = 0.7167630057803468 **F1 = 0.8169804301763245**
Max $\boldsymbol{w_x}$	Fn= 0.8975514201762977 Fa = 0.8531211750305998 Fo= 0.7485613103142984 **F1= 0.8330779685070654**	Fn = 0.8969849246231156 Fa = 0.8444444444444444 Fo = 0.7315634218289085 **F1 = 0.824330930298823**
Logistic Regression 6	Fn= 0.9031496062992126 Fa = 0.867696440564137 Fo= 0.7672395501803522 **F1= 0.8460285323479005**	Fn = 0.8981132075471698 Fa = 0.8395061728395061 Fo = 0.7206703910614525 **F1 = 0.8194299238160427**
Logistic Regression 9	Fn = 0.902857142857142 Fa = 0.870946393117141 Fo = 0.767496277387789 **F1 = 0.847099937787357**	Fn = 0.8978562421185372 Fa = 0.8433734939759037 Fo = 0.7247191011235955 **F1 = 0.8219829457393454**
Logistic Regression 21	Fn = 0.904284591194968 Fa = 0.871523178807947 Fo = 0.773310521813515 **F1 = 0.849706097272143**	Fn = 0.8938826466916354 Fa = 0.8484848484848485 Fo = 0.7163323782234957 **F1 = 0.8195666244666598**
Random Forest 6	Fn = 0.903782088967274 Fa = 0.8728246318607764 Fo = 0.767481767481767 **F1 = 0.848029496103272**	Fn = 0.8986232790988736 Fa = 0.8536585365853658 Fo = 0.7215909090909091 **F1 = 0.8246242415917161**
Random Forest 9	Fn = 0.905056455571919 Fa = 0.868632707774799 Fo = 0.771824973319103 **F1 = 0.848504712221940**	Fn = 0.8981132075471698 Fa = 0.8484848484848485 Fo = 0.7211267605633803 **F1 = 0.8225749388651328**
Random Forest21	Fn = 0.905060808160062 Fa = 0.869031377899045 Fo = 0.774042553191489 **F1 = 0.849378246416865**	Fn = 0.8905660377358491 Fa = 0.8518518518518519 Fo = 0.7094972067039106 **F1 = 0.8173050320972038**
Support Vector Machine 6	Fn = 0.904887328065554 Fa = 0.8735936465916612 Fo = 0.768260869565217	Fn = 0.8977556109725686 Fa = 0.8433734939759037 Fo = 0.7204610951008645

	F1 = 0.848913948074144	**F1 = 0.8205300666831122**
Support Vector Machine 9	Fn = 0.902336494280965 Fa = 0.870527000650618 Fo = 0.761966927763272 **F1 = 0.844943474231618**	Fn = 0.8958594730238394 Fa = 0.8588235294117647 Fo = 0.7241379310344828 **F1 = 0.8262736444900289**
Support Vector Machine 21	Fn = 0.902271837054445 Fa = 0.871261378413524 Fo = 0.763226366001734 **F1 = 0.845586527156568**	Fn = 0.8969849246231156 Fa = 0.8588235294117647 Fo = 0.7277936962750716 **F1 = 0.8278673834366507**
Naive Bayes 6	Fn = 0.893704647970190 Fa = 0.8589661774090619 Fo = 0.735262127474439 **F1 = 0.829310984284564**	Fn = 0.89281210592686 Fa = 0.8554913294797688 Fo = 0.7048710601719198 **F1 = 0.8177248318595162**
Naive Bayes 9	Fn = 0.896300669027941 Fa = 0.856962822936358 Fo = 0.743873346345695 **F1 = 0.832378946103331**	Fn = 0.8964646464646465 Fa = 0.8457142857142858 Fo = 0.7183908045977011 **F1 = 0.8201899122588778**
Naive Bayes 21	Fn = 0.895945285784074 Fa = 0.8550093341630367 Fo = 0.736283185840708 **F1 = 0.829079268595939**	Fn = 0.8944723618090452 Fa = 0.8539325842696629 Fo = 0.7096774193548387 **F1 = 0.8193607884778489**