https://orcid.org/0000-0003-2204-0487
https://orcid.org/0000-0002-5232-9487
https://orcid.org/0000-0002-0195-2770
https://orcid.org/0000-0002-2300-6320

УДК 004.896

В.В. ВИШНЕВСЬКИЙ[*], Т.М. РОМАНЕНКО[*], Ю.О. ЛУГОВСЬКИЙ[*], О.Ф. БОРЕЦЬКИЙ[**]

СЕРВІСИ ПІДГОТОВКИ ДАНИХ ДЛЯ АВТЕНТИФІКАЦІЇ ЛЮДИНИ ЗА ЇЇ ЕЛЕКТРОКАРДІОГРАМОЮ

[*]Інститут проблем математичних машин і систем НАН України, м. Київ, Україна
[**]Київський національний університет імені Тараса Шевченка, м. Київ, Україна

Анотація. *Для сучасної телемедицини актуальним є вирішення задачі автентифікації пацієнта. У цій роботі досліджуються можливості використання особистих унікальних ознак людини, а саме її електрокардіограми (ЕКГ), для автентифікації. Біометрична автентифікація є перспективною в телемедицині, де все більші обсяги даних пацієнт надсилає лікарям інформаційними каналами. Коли клініка спостерігає за своїм пацієнтом дистанційно та просить надіслати його ЕКГ, вона має бути впевнена, що отримана ЕКГ належить саме цьому пацієнтові. Біометрична автентифікація з використанням ЕКГ – це доволі специфічний напрям дослідження і наразі ще не існує провідних технологій у цій сфері. Така біометрична автентифікація і є метою технології, що розробляється. У даному напрямі нами вже були проведені дослідження, розроблені алгоритми та проведена перевірка їх дієздатності на тестових даних. Об'єднавши отримані алгоритми та машинне навчання для вирішення задачі класифікації приналежності ЕКГ людині, отримали програму, здатну автентифікувати людей за ЕКГ. Це дало впевненість у можливості використання такої технології. Наступним кроком стало масштабування цієї технології задля двох цілей: перевірити працездатність технології на значно більшій кількості даних та підготувати її для використання в реальних умовах, коли час на обробку має бути мінімальним. У статті описується поетапний розвиток від програми, що запускалась на персональному комп'ютері, до системи, яка здійснює обчислення у хмарному середовищі. Показані затрати часу при проведенні експериментів і виконане порівняння часу роботи програм на персональному комп'ютері та у хмарному середовищі, експериментальним шляхом знайдена найефективніша конфігурація системи для хмарної архітектури.*
Ключові слова: *електрокардіограма, класифікація, автентифікація, хмарна архітектура.*

Abstract. *For modern telemedicine, it is important to solve the problem of patient authentication. This paper studies the possibility of using personal unique features of people, namely their electrocardiogram (ECG), for authentication. Biometric authentication is promising in telemedicine, where patients send physicians through information channels increasing amounts of their data. When some clinic monitors its patient remotely and asks them for an ECG, it should be confident that the received ECG belongs to that particular patient. Biometric authentication using ECG is a rather specific area of research and now there are no leading technologies in this field. This method is the goal of the technology being developed. In this direction, we have already conducted research, developed algorithms, and tested their viability on test data. Combining the obtained algorithms and machine learning to solve the problem of classification of ECG belonging to a person, there was developed a program capable of authenticating people by their ECG. This gave confidence in the possibility of using such technology. The next step was to scale this technology to reach two goals: to test the performance of the technology on a much larger amount of data and to prepare it for use in real conditions when processing time should be minimal. The paper describes the gradual development from a program running on a personal computer to a system performing computing in a cloud environment. The time spent on conducting experiments and comparing the running time of programs on a personal computer and in a cloud environment are shown. The most efficient configuration of the system for cloud architecture was found experimentally.*
Keywords: *electrocardiogram, classification, authentication, cloud architecture.*

DOI: 10.34121/1028-9763-2022-3-58-69

1. Вступ

Будь-яка сучасна технологія телемедицини стикається з необхідністю підтвердження особи пацієнта, який знаходиться на відстані від лікаря. Найбільш поширені на цей час технології віддаленого аналізу ЕКГ не є винятком. Оскільки предметом такого телемедичного консультування є сигнал, що реєструється безпосередньо з людини, представляється виграшним вирішення задачі підтвердження особи пацієнта за рахунок розробки відповідної інтелектуальної технології аналізу цього сигналу. В галузі кіберзахисту даних такий підхід має назву біометричної автентифікації.

Хоча цей напрям досліджений ще недостатньо, однак вже має певний комерційний інтерес. Наприклад, потенційним споживачем такої технології може бути страхова медицина, яка поступово діджиталізується і починає працювати із клієнтами дистанційно.

Використання параметрів ЕКГ для біометричної автентифікації є відносно новим напрямом, тому нині ще не існує технології, яка б лідирувала і масово використовувалась для автентифікації великої кількості людей. Підходи в розпізнаванні ЕКГ дуже різні. У публікаціях [1–5] можна побачити, що автори досліджують можливість розпізнавання сигналів ЕКГ за їхніми інтервальними та амплітудними характеристиками, морфологією серцебиття, фазовими портретами тощо.

У наших попередніх роботах [6–11] наведено, як за допомогою кардіосигналу людини можна здійснювати її біометричну автентифікацію. На цей оригінальний спосіб автоматичної автентифікації людини за її багатоканальною ЕКГ отримано патент [12].

Метою цієї статті є дослідження і застосування розроблених алгоритмів у програмних рішеннях, що використовуються для розв’язання задач автентифікації і побудови на їхній основі частини системи масової обробки ЕКГ.

2. Постановка задачі

Щоб досягнути поставленої мети, було потрібно провести велику кількість дрібних досліджень, розробити прототипи та модернізувати їх. Першим прототипом стала демонстраційна програма з відповідними бібліотеками, які використовують у своїй роботі алгоритми автентифікації, що описані у статтях [6–10]. За допомогою цієї програми проводилась перевірка розроблених алгоритмів. Після цього стали зрозумілими задачі, які необхідно вирішити.

Демонстраційна програма здійснювала такі операції: обробку вхідних даних ЕКГ, обчислення кривих, підготовку даних для навчання нейромереж, навчання нейромереж, зберігання нейромереж і використання їх для автентифікації конкретної людини. За її допомогою переконалися, що алгоритми автентифікаціїї працездатні. Однак, щоб упевнитися в універсальності їх роботи, потрібно далі провести дослідження на великій кількості ЕКГ різних людей. Використання лише цієї демонстраційної програми робить такий процес дуже рутинним і надто затратним у часі, оскільки вона не має інструментів для одночасної обробки даних.

Для того, щоб переконатись, що алгоритми працюють і для великої кількості осіб, потрібно інтегрувати їх в іншу програму, яка буде здатною обробляти велику кількість даних одночасно. Отримане рішення дозволить не тільки проводити ефективніше подальші дослідження, а й матиме практичне застосування в реальній системі по масовій автентифікації.

Таким чином, мають бути зроблені такі кроки для досягнення бажаних цілей:

• Розробити програму, яка на основі даних електрокардіограм навчить нейромережі, необхідні для алгоритму автентифікації. Навчання будуть відбуватись послідовно, але повністю в автоматизованому режимі. Розроблена програма має бути побудована таким

чином, щоб у подальшому її легко було переробити для одночасних обчислень різних електрокардіограм.

• Налаштувати сервіси приватного хмарного середовища.

• Запустити розроблений додаток у хмарному середовищі. Це дозволить підготувати базу для хмарних обчислень.

• Розробити і підготувати хмарну архітектуру, яка одночасно оброблятиме ЕКГ різних людей. Визначити максимально ефективну конфігурацію архітектури, при якій одночасне навчання нейромереж займатиме найменше часу.

Поставлені цілі стосуються лише тієї частини робіт, що відповідають за обробку електрокардіограм, підготовку даних і навчання нейромереж. Сюди не входить одночасна перевірка роботи автентифікації. Автоматизація цього процесу буде виконана таким чином.

Далі будуть описані технологія автентифікації та дані, які вона використовує, також будуть наведені архітектурні рішення й описана їхня ефективність.

3. Технологія автентифікації однієї особи

Автентифікація людини виконується у два етапи.

На першому етапі проводиться навчання нейромережі, а на другому – безпосередньо автентифікація з використанням навчених нейромереж. Нейромережі для конкретної особи навчаються заздалегідь і можуть бути багаторазово використані для її автентифікації.

Даними для навчання нейромереж є еталонна ЕКГ, а саме фільтрований сигнал та файл розмітки особи, що має бути автентифікована. Також необхідними є підготовлені заздалегідь дані ЕКГ деякої кількості допоміжних персон. Ці допоміжні персони є однаковими для всіх осіб, що будуть автентифікуватися.

Еталонна ЕКГ людини, яку треба автентифікувати, обробляється у відповідності з алгоритмом, детально описаним у [6, 8, 11]. Коротко, послідовність дій є такою: з файла фільтрованого сигналу потрібно вирізати QRS-комплекси, використовуючи файл розмітки, сформувати тривимірні образи QRS-комплексів, ранжувати їх відповідно до метрик Хаусдорфа, обрати певну кількість образів для навчання нейромережі, апроксимувати обрані образи канонічними сплайнами у тривимірному просторі координат (4 контрольні точки). Ці 12 координат контрольних точок будуть вектором ознак для навчання нейромережі.

Далі виконується підготовка даних для побудови нейромереж, навчених на розпізнавання двох класів: першим класом у кожній парі є дані, отримані з еталонної ЕКГ особи, що автентифікується, а другим класом – дані однієї з допоміжних персон.

Наступний крок – побудова навчених на два класи нейромереж та зберігання їх у відповідних каталогах.

Для другого етапу автентифікаціїї використовуються дані поточної ЕКГ (фільтрований сигнал і файл розмітки), тобто тієї ЕКГ, приналежність якої до певної особи треба підтвердити. Порядок обробки поточної ЕКГ є аналогічним описаній вище обробці еталонної ЕКГ, тільки для автентифікації обирається один QRS-комплекс, для якого метрика Хаусдорфа буде найменшою (репрезентативний QRS-комплекс). Після його апроксимації канонічними сплайнами у тривимірному просторі отримуємо 12 координат контрольних точок, які є вектором ознак для класифікації.

Далі проводимо класифікацію отриманого вектора ознак на всіх навчених нейромережах. У випадку позитивної відповіді про приналежність поточної ЕКГ особі, що автентифікується більш, ніж у 70 відсотках навчених нейромереж, приймаємо рішення про її дійсну приналежність цій особі.

Для виконання описаного алгоритму розроблені дві бібліотеки: бібліотека обробки ЕКГ та бібліотека навчання нейромереж.

Бібліотека обробки ЕКГ виконує вирізання QRS-комплексів із фільтрованого сигналу ЕКГ за допомогою файла розмітки, обчислює Хаусдорфові відстані між вирізаними QRS-комплексами, ранжує QRS-комплекси у відповідності з Хаусдорфовими відстанями, обирає певну кількість QRS-комплексів із меншими значеннями цих відстаней для навчання нейромереж або один QRS-комплекс із найменшим значенням відстані Хаусдорфа для класифікації, апроксимує обраний QRS-комплекс канонічними сплайнами у тривимірному просторі координат, отримуючи 12 координат контрольних точок, які складатимуть вектор ознак для класифікації нейромережею.

Бібліотека навчання нейромереж використовує для навчання дані, отримані в результаті роботи бібліотеки обробки ЕКГ. Бібліотека написана таким чином, що для навчання приймає тривимірні координати, які описують криву будь-якою кількістю точок. На даному етапі створюються криві, які описуються чотирма контрольними точками.

Винесення бібліотек в окремі проєкти дозволило створити з них nuget-пакети і завантажити їх до системи контролю версій GitLab. Тепер ці бібліотеки можна включати в будь-який інший проєкт, де потрібно здійснювати подібні обчислення. Таке рішення дозволяє тримати алгоритми знаходження тривимірних координат та навчання нейромереж в одному місці, що запобігає дублюванню коду та пов’язаних із цим інших проблем.

4. Алгоритми і архітектурні рішення. Десктоп

Нижче описуються програма та її алгоритм, який здійснює автоматизоване навчання для багатьох персон.

Розроблено консольний застосунок на платформі .net 5, написаний на мові програмування C#. Перевагою .net 5 є кросплатформеність, тобто такий застосунок можна запускати на інших операційних системах. Цей застосунок обробляє файли з сигналом ЕКГ та по отриманих даних навчає нейромережу, яка зберігається у бінарному файлі. Виконання консольного застосунку є проведенням експерименту. Для кожного експерименту створюється окрема папка, в якій зберігаються файли ЕКГ, що є вхідними даними, а також створюється окрема папка з результатами обробки ЕКГ та навченими нейромережами. Завдяки цьому досягається ізольованість експерименту, де вхідні і вихідні дані знаходяться в конкретних папках експерименту. Навчені моделі можна використовувати для автентифікації.

Алгоритм роботи програми наведений на рис. 1.

1. Вхідною інформацією для програми є ім’я експерименту, що, у свою чергу, є назвою папки, в якій зберігаються файли розмітки та фільтрованого сигналу (дані ЕКГ) для експерименту. Програма отримує назви файлів, формує список пар і послідовно, для кожної пари, виконує цикл, що закінчується навчанням і збереженням нейромережі. Кількість таких циклів залежить від кількості персон, для яких підготовлені такі файли.

2. Файли з даними ЕКГ завантажуються в оперативну пам’ять.

3. На основі файлів із даними ЕКГ бібліотечними функціями обробки ЕКГ обчислюються координати контрольних точок сплайнів, за допомогою яких ЕКГ представляється у тривимірному просторі координат.

4. Завантажуються координати контрольних точок сплайнів, за допомогою яких представляються ЕКГ допоміжних персон у тривимірному просторі координат.

5. З координат особи, що автентифікуватиметься, та допоміжних персон формується файл, який використовуватиметься для навчання нейромережі.

6. Файл для навчання нейромереж зберігається у файловій системі.

7. Файл для навчання нейромереж завантажується з файлової системи.

8. Файл для навчання обробляється, і отримується певна кількість наборів даних, де в кожному наборі є координати однієї особи, що автентифікуватиметься, і однієї допоміжної персони. Для кожної пари запускається цикл навчання нейромережі.

9. Дані для навчання передаються бібліотеці машинного навчання. В даному випадку передаються чотири контрольні точки для опису сплайна.

10. Навчена нейромережа зберігається у файловому сховищі.

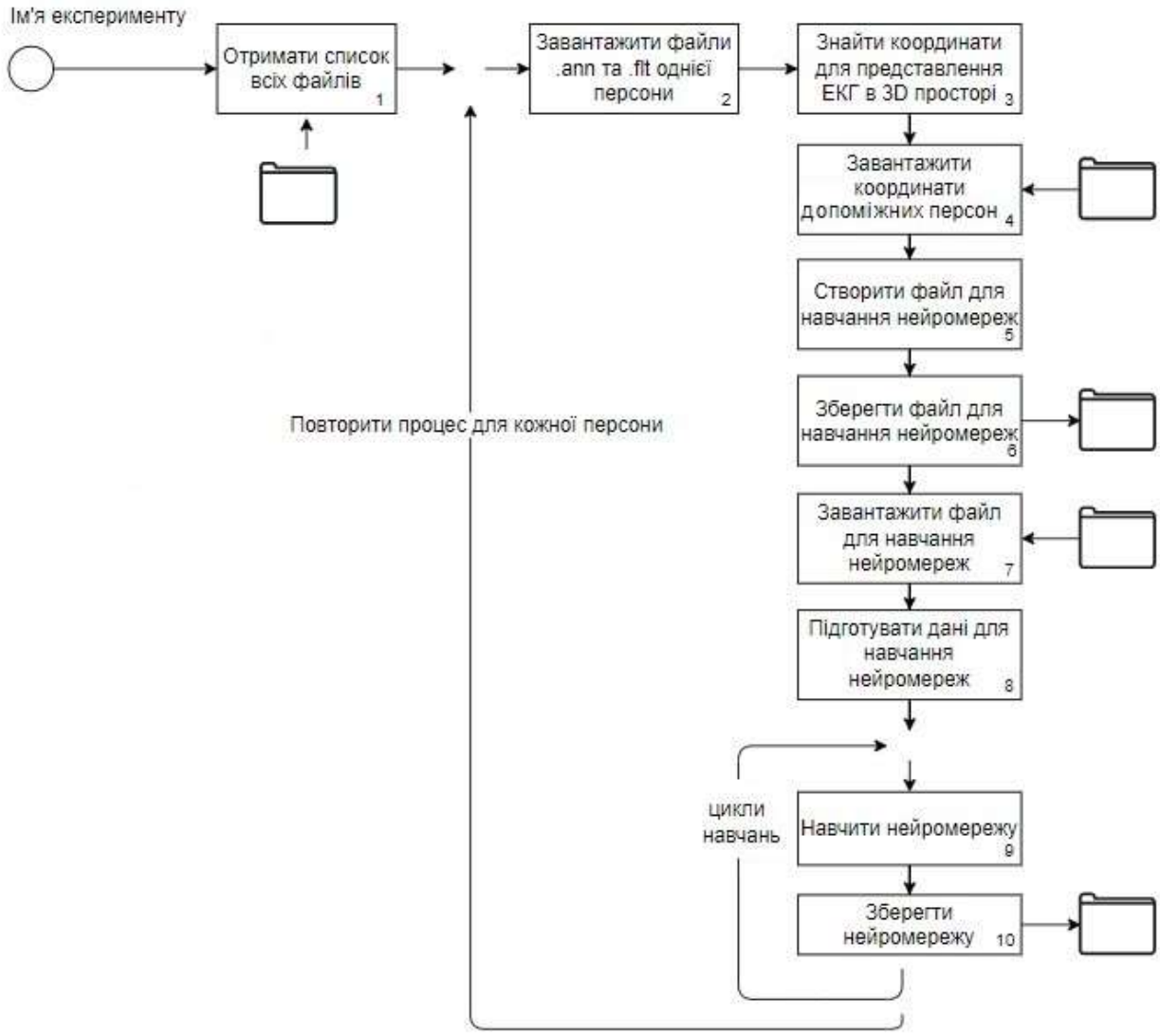

Рисунок 1 – Алгоритм роботи програми

Послідовний процес побудови нейромереж працює, але, в разі проведення експериментів з великою кількістю людей, займатиме багато часу. Це пов’язано з великою кількістю обчислень, необхідних для побудови кожної нейромережі. Часові затрати будуть описані у подальших розділах цієї статті.

Для усунення даної проблеми потрібно робити обчислення паралельно для кожної персони. Цього можна досягнути, переробивши додаток у WEB-сервіс та розгорнувши його у хмарному середовищі, де доступ до нього здійснюватиметься через HTTP-запит. Оскільки це WEB-сервіс, то він зможе приймати одночасно декілька запитів на обчислення і здійснювати їх паралельно. Для кращої масштабованості варто винести тривалі за часом операції в окремі WEB-сервіси. Така архітектура називається мікросервісною, тому що для

кожної функціонально незалежної одиниці створюється свій WEB-сервіс. Така мікросервісна архітектура дозволить добре розпаралелювати обчислення.

5. Приватні хмарні сервіси, що задіяні в технології

Приватна хмарна інфраструктурі проєкту “Медгрід” побудована на основі програмного комплексу керування хмарною інфраструктурою OpenStack. У цій хмарній інфраструктурі розгорнуто систему оркестрації контейнерів Kubernetes як середовище запуску мікросервісів [13]. Для зазначених у статті експериментів було використано мінімальну з точки зору адміністрування та обслуговування конфігурацію з одним вузлом кластера, що було запущено як віртуальну машину у OpenStack. На окремій віртуальній машині розгорнуто S3-сумісний сервіс зберігання даних MinIO, який дозволяє зберігати дані, що використовуються у роботі мікросервісів.

Також, із метою систематизації розробки програмних засобів у проєкті “Медгрід” розгорнуто програмний комплекс GitLab. Даний ресурс дозволяє забезпечити збереження коду розроблюваних сервісів у проєкті, організації процесу розробки програмних засобів та ведення внутрішньої технічної документації програмних засобів. На основі платформи GitLab реалізовано інфраструктуру для збірки та розгортання мікросервісів у Kubernetes [13]. У програмному комплексі присутній репозиторій образів, який використовується для збереження контейнерів мікросервісів автентифікації.

Мікросервіси розміщуються в окремих репозиторіях, які містять сценарій збірки образу контейнера його розміщення у репозиторії образів та розгортання нової версії мікросервісу у Kubernetes і конфігурації інфраструктури для роботи мікросервісу.

У результаті застосування в інфраструктуру проєкту “Медгрід” вищезазначених елементів отримано взаємодію, показану на рис. 2. Такий підхід дозволив зменшити кількість дій від виконавців проєкту, необхідних при розробці елементів сервісу автентифікації, та зменшити час від розміщення нової версії коду мікросервісу у репозиторії до працюючого мікросервісу, розгорнутого у Kubernetes кластері до декількох хвилин.

6. Алгоритми і архітектурні рішення. Хмара

Нижче наведено, як консольний застосунок був перероблений у хмарне рішення, що дозволяє одночасно обробляти дані декількох персон.

Програма, яка описана вище, перенесена у хмарне середовище з розбиттям її на декілька мікросервісів. Схема розбиття наведена на рис. 2.

Утворено 4 мікросервіси:

1. EcgAuth.ExperimentRunnerApi. Мікросервіс, що керує експериментом. Він бере на себе відповідальність у визначенні, для яких саме персон відбуватиметься обчислення координат, необхідних для навчання нейромереж, і навчання самих нейромереж.

2. EcgAuth.EcgProcessingApi. Задачею цього мікросервісу є підготовка файла для навчання нейромережі. Він, за допомогою бібліотеки EcgAuth.EcgProcessing обчислює координати для представлення ЕКГ у тривимірному просторі координат. До цих координат додає координати допоміжних персон і зберігає у файлі для навчання нейромережі.

3. EcgAuth.LearningRunnerApi. Задачею мікросервісу є підготовка наборів даних, на яких навчатимуться нейромережі на основі файла для навчання нейромережі. У кожному наборі даних будуть координати однієї особи, що автентифікується, і однієї допоміжної персони.

4. EcgAuth.MachineLearning.EngineApi. Задачею мікросервісу є навчання нейромережі на основі переданих даних і збереження її для подальшого використання.

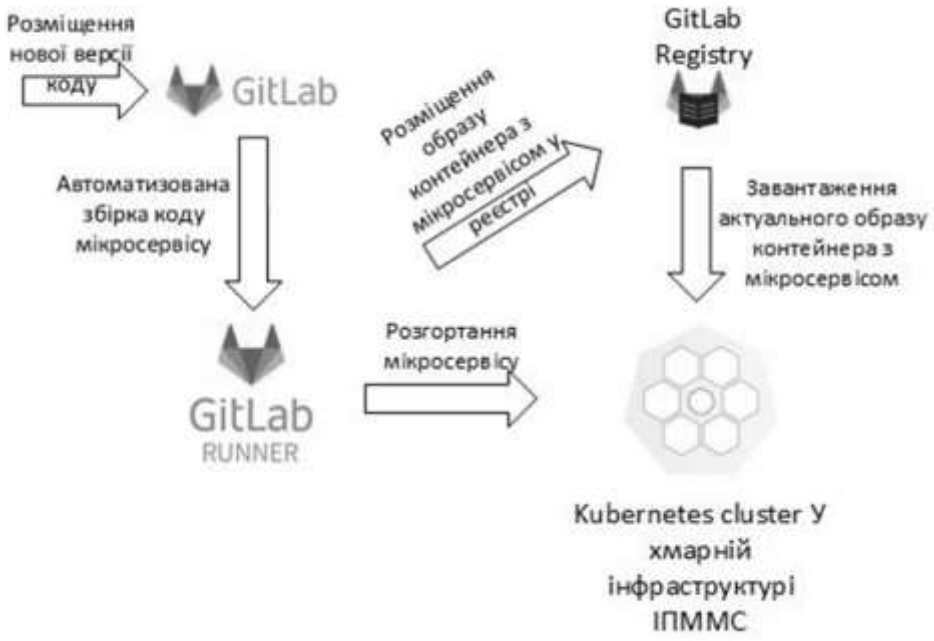

Рисунок 2 – Взаємодія елементів приватної хмарної інфраструктури проєкту “Медгрід”

У цій мікросервісній архітектурі повністю повторений принцип роботи з файлами. Тільки файли зберігаються не в локальній папці у комп’ютері, а у хмарному сховищі.

На рис. 3 наведена схема взаємодії описаних мікросервісів між собою та файловим сховищем.

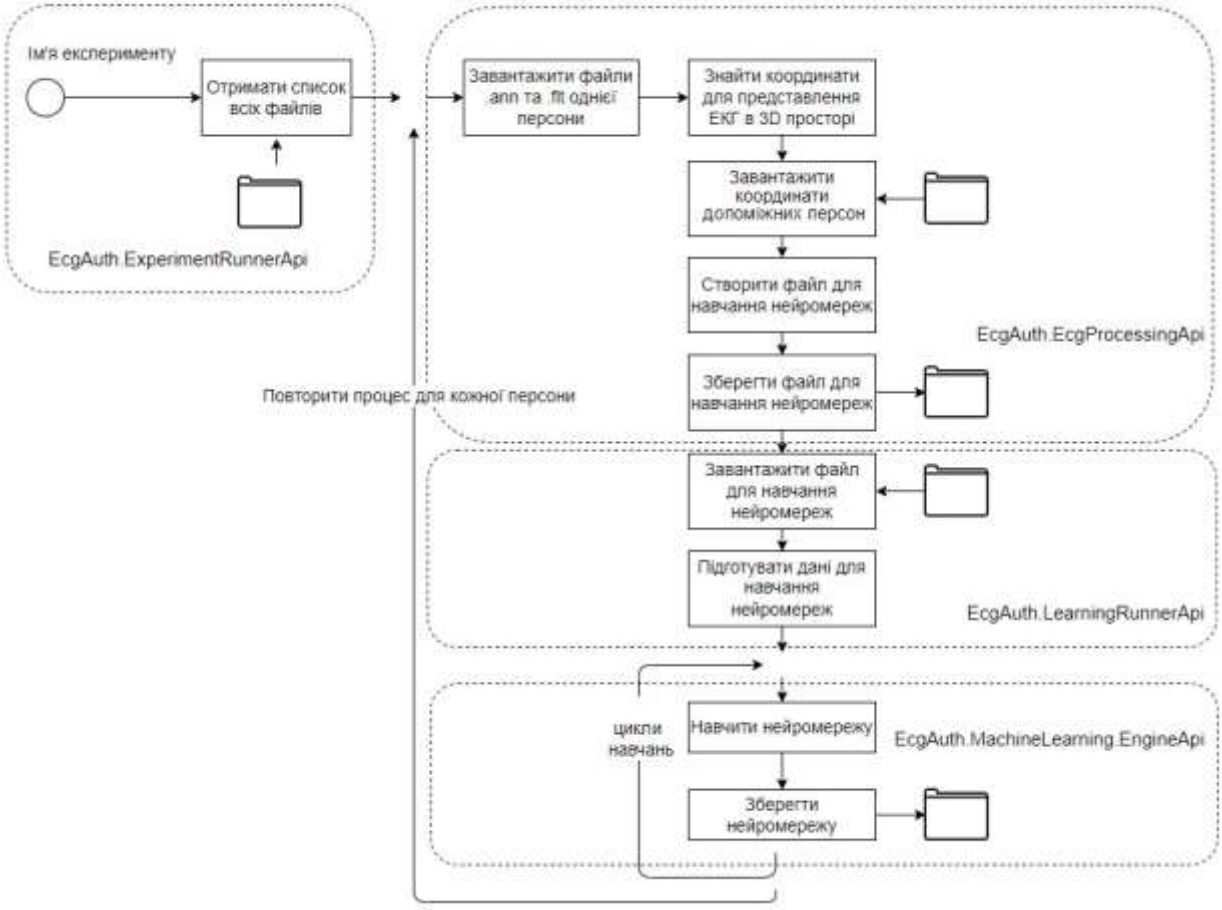

Рисунок 3 – Схематичний поділ програми на окремі мікросервіси

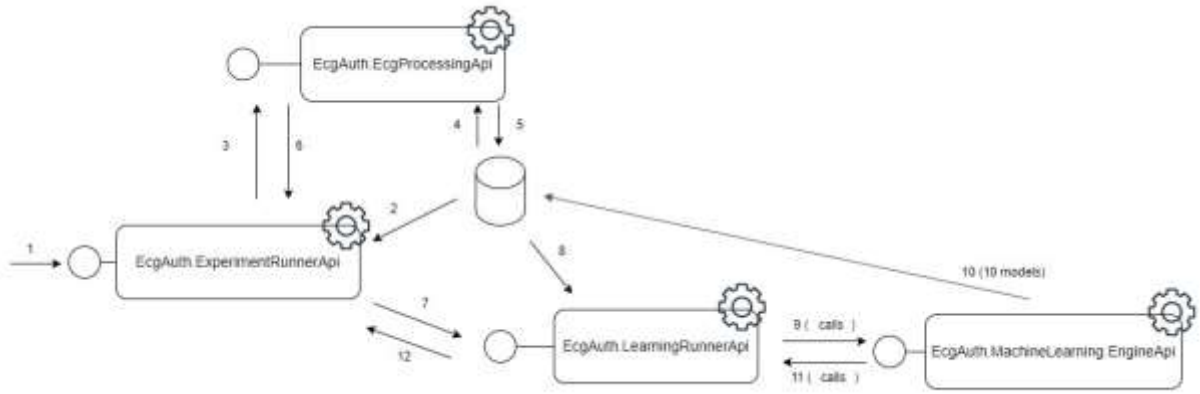

Рисунок 4 – Схема взаємодії мікросервісів між собою та файловим сховищем

На рис. 3 циліндр представляє собою хмарне сховище для файлів, розгорнуте у хмарному середовищі з використанням продукту MinIO. MinIO – це високошвидкісне сховище для зберігання об'єктів. У цьому сховищі створюються окремі папки, в яких зберігаються файли. Ці папки називають бакетами (від англ. bucket). В одного бакета може бути будь-яка кількість вкладених бакетів.

Якщо розглянути комунікацію між мікросервісами, то запити надсилаються в такому порядку.

1. Запит на виконання експерименту. Його отримує мікросервіс EcgAuth.ExperimentRunnerApi. Як вхідний параметр передається назва бакета, в якому зберігаються файли даних ЕКГ (фільтрований сигнал та файл розмітки). Тобто назва експерименту і бакета мають збігатися.

2. З бакета зчитується список імен наявних файлів для експерименту.

3. Запит на обробку однієї пари файлів даних надсилається мікросервісу EcgAuth.EcgProcessingApi.

4. Мікросервіс EcgAuth.EcgProcessingApi завантажує файли даних із бакета і починає їх обробку. Після завершення обчислення координат для тривимірної кривої ЕКГ завантажуються координати допоміжних персон із бакета і формується файл для навчання нейромережі.

5. Збереження створеного файла для навчання нейромережі.

6. Повертається відповідь від мікросервісу EcgAuth.EcgProcessingApi. Повертається Guid, що представляє собою ідентифікатор персони, для якої було створено файл для навчання нейромережі. Цей Guid є ідентифікатором персони в нашій системі.

7. Запит на навчання нейромережі для конкретної персони надсилається мікросервісу EcgAuth.LearningRunnerApi. Параметром передається Guid персони та ім'я експерименту. Після цього Guid шукатиме файл для навчання нейромережі в бакеті експерименту.

8. EcgAuth.LearningRunnerApi завантажує файл для навчання нейромережі для конкретної персони. З цього файла мікросервіс створює набори даних, які передаватимуться для навчання нейромережі.

9. Запит на навчання нейромережі для конкретної пари надсилається мікросервісу EcgAuth.MachineLearning.EngineApi. Параметрами передається Guid персони, ім'я експерименту та дані, на яких навчатиметься нейромережа. Ідентифікатор персони та ім'я експерименту потрібно, щоб зберегти створені для персони нейромережі в потрібному місці.

10. EcgAuth.MachineLearning.EngineApi зберігає навчену мережу у бакеті.

11. Відповідь мікросервісу EcgAuth.LearningRunnerApi про завершення навчання нейромережі для одного набору даних.

12. Відповідь мікросервісу EcgAuth.ExperimentRunnerApi про завершення створення моделей для конкретної персони.

Завдяки запропонованій мікросервісній архітектурі можна зменшити сумарний час обчислень за рахунок розпаралелювання роботи мікросервісів, що є найбільш навантаженими.

Дану архітектуру найбільш доцільно розгорнути на інфраструктурі, розрахованій на роботу мікросервісів. Тому для її тестування та відлагодження достатньо використати один вузол кластера Kubernetes, який не буде задіяно для інших задач. Результати представлені в наступному розділі.

7. Експерименти

У даному розділі представлена послідовність проведених експериментів. По них можна відслідкувати, скільки часу було використано на обробку і навчання нейромереж одного й того ж набору даних. У ході експериментів визначились найефективніші параметри для розгорнутої архітектури.

Перший експеримент проводився на персональному комп'ютері з використанням консольного застосунку. Обсяг вхідних даних – 30 персон. У консольному застосунку всі операції обчислення і навчання нейромереж відбувались послідовно. Навчання нейромереж здійснювалось бібліотекою машинного навчання ML.NET. Ця бібліотека використовувала всі доступні ядра персонального комп'ютера для навчання однієї нейромережі.

Консольний застосунок запускався на двоядерному процесорі з тактовою частотою 2.3 ГГц. Вимірювання часу показало, що в середньому цикл обробки даних однієї персони з моменту завантаження файлів ЕКГ до моменту збереження нейромережі в бінарному файлі займав приблизно 22–26 с. Найбільш тривалим був процес навчання нейромереж. Він, у середньому, займав приблизно 17–20 с. Іншим затратним щодо часу місцем програми було знаходження координат, що описують ЕКГ у тривимірному просторі і яке склало 4-6 с. Таким чином, в експерименті для обчислення даних 30 персон та навчання нейромереж знадобилося 12 хв і 41 с.

Однією з цілей запуску алгоритмів у хмарі було розгорнути консольний застосунок, написаний на мові C#, на віртуальній машині у хмарі, що використовує операційну систему LinuxЄ і провести там експеримент. Консольний застосунок було розгорнуто на віртуальній машині, під яку виділили 2 фізичних ядра процесора. Експеримент проводився на тих же 30 персонах, і загальний час виконання склав 11 хвилин 46 секунд. Це майже на хвилину швидше в порівнянні з персональним комп'ютером. Цим експериментом досягнута мета запуску розроблених алгоритмів у хмарі. Також ми переконалися в достатності обчислювальних потужностей для проведення подальших експериментів у хмарі.

Наступною метою стало вирішення питання, наскільки мікросервісна архітектура краща за консольний застосунок. Тому на аналогічній віртуальній машині, яка використовувалась як вузол Kubernetes, було розгорнуто по одному екземпляру мікросервісів і проведений аналогічний експеримент, де всі обчислення теж відбуваються послідовно. Тобто, мікросервіс, відправивши запит на інший мікросервіс, чекав відповідь і тільки потім надсилав наступний запит. Загальний час послідовних обчислень склав 12 хв і 8 с. Це на 32 с довше за роботу консольного застосунку на віртуальній машині. В цьому є сенс, тому що йшли додаткові часові затрати на пересилання HTTP-запитів між мікросервісами і кожен мікросервіс здійснював додаткові обчислення з обробки запитів. У цьому випадку, коли всі обчислення відбуваються послідовно, мікросервісна архітектура

не є кращим рішенням. Однак, у даному експерименті ігнорувалась основна здатність мікросервісів опрацьовувати 2 і більше запитів одночасно. Якщо розглянути результати експерименту з консольним застосунком, то приблизно 75% всього часу йде на навчання нейромережі. Тому в наступних експериментах одночасно відправляли 2 і більше запитів на мікросервіс, що навчає нейромережі.

Проводились подальші експерименти на тих самих 30 персонах, але водночас навчалось 2 і більше нейромереж. Попередні і наступні результати наведені в табл. 1.

Таблиця 1 – Результати експериментів на вузлі Kubernetes із двома виділеними ядрами процесора

Одночасно навчалось нейромереж	Загальний час обчислень 30 персон	Середній час навчання однієї нейромережі
1	12 хв 8 с	2,1 с
2	10 хв 31 с	3,2 с
3	9 хв 2 с	4 с
4	8 хв 59 с	5,2 с
5	8 хв 47 с	6,4 с

У табл. 1 приведений також середній час навчання нейромережі. Помітно, що зі збільшенням кількості нейромереж, які навчаються одночасно, збільшується середній час на навчання нейромережі. Це пов’язане з тим, що процесор постійно переключається між навчаннями різних нейромереж, і тому час навчання однієї нейромережі зростає. Також помітно, що немає сенсу навчати більше 5 нейромереж одночасно, тому що загальний час обчислень вже майже не зменшується. За цих умов процесор завантажений по максимуму.

Наступним кроком стала спроба ще покращити ці результати, виділивши додаткові ядра процесора для вузла Kubernetes. Спочатку додали 2 ядра і провели такі ж експерименти на 30 персонах. Потім додали ще 2 ядра, збільшивши їх сумарну кількість до 6. І останні експерименти провели, виділивши 8 ядер під вузол Kubernetes. Результати експериментів наведені у табл. 2.

Таблиця 2 – Результати експериментів на вузлі Kubernetes із різною кількістю виділених ядер процесора

Кількість виділених ядер	Одночасно навчалось нейромереж	Загальний час обчислень 30 персон	Середній час навчання однієї нейромережі
4	1	12 хв 31 с	2,2 с
4	2	8 хв. 15 с	2,7 с
4	3	7 хв 51 с	3.5 с
4	4	7 хв 19 с	4,1 с
4	5	6 хв 59 с	5 с
6	1	15хв 28 с	2,8 с
6	2	9 хв 39 с	3,2 с
6	3	8 хв 52 с	3,9 с
6	4	8 хв 48 с	4,7 с
8	1	20 хв 3 с	3,7 с
8	2	12 хв 18 с	4,2 с
8	3	10 хв 29 с	4,7 с
8	4	9 хв 33 с	5,8 с

Найкращі результати були отримані при виділенні чотирьох фізичних ядер процесора під вузол Kubernetes, при цьому одночасно навчалося 5 нейромереж. Можна вважати, що для даної архітектури, де всі мікросервіси розгорнуті на одному вузлі Kubernetes і їм доступні всі виділені ядра машини, така конфігурація є найкращою.

Аналізуючи результати, отримані при розгортанні мікросервісів на вузлі Kubernetes, коли доступні 6 і 8 ядер, бачимо, що загальний час обчислень збільшується. Це пов'язано зі збільшенням середнього часу на навчання однієї нейромережі. Якщо порівняти середній час при послідовному обчисленні на експерименті з виділеними двома ядрами (2,1 с) і вісьмома (3,7 с), то час збільшився на 1,6 с. Це пов'язано з тим, що мікросервіс, який навчає нейромережу, використовує бібліотеку ML.NET і вона під час навчання однієї нейромережі намагається використати всі доступні ядра процесора і робить це неефективно. Тому виділяти велику кількість ядер під мікросервіс, що навчає нейромережі, не варто.

8. Подальші дослідження

Отримані результати експериментів вказують на те, що виділення великої кількості ядер під вузол Kubernetes, де навчаються нейромережі, не є найкращим рішенням. Враховуючи особливість, що зі збільшенням кількості ядер час на навчання нейромережі збільшується, варто розгорнути мікросервіс, який навчає нейромережу, на окремих вузлах Kubernetes, де кожній буде виділено одне ядро фізичного процесора. Під час навчання однієї персони слід одразу надсилати запити на навчання нейромережі на різні мікросервіси. Завдяки такому рішенню при навчанні всіх нейромереж одночасно середній час на навчання нейромережі залишатиметься близько 2 с. Це дозволить значно скоротити час на підготовку даних для автентифікації однієї персони.

Далі стоятиме задача оптимізації обчислень для знаходження тривимірних координат, що використовуються для навчання нейромережі.

9. Висновки

Виконана робота свідчить, що кроки з підготовки алгоритмів автентифікації і виведення їх в масове використання частково зроблені.

Найефективнішою архітектурою для підготовки даних для майбутньої автентифікації виявилась мікросервісна архітектура, розгорнута на вузлі Kubernetes із виділеними для нього чотирма ядрами процесора, де одночасно навчалось 5 нейромереж. У цьому випадку сумарний час на обчислення даних багатьох персон найменший.

Однак експерименти на вузлі Kubernetes при виділенні шести і восьми ядер процесора показують, що отримане рішення в цих експериментах не є найкращим і варто продовжувати змінювати архітектуру. Виділення окремих вузлів Kubernetes кластера для кожного екземпляра мікросервісу, що навчає нейромережі, буде кращим, і при цьому час підготовки даних для автентифікації нової персони буде мінімальним.

СПИСОК ДЖЕРЕЛ

1. Yogendra N.S., Gupta P. Biometrics Method for Human Identification Using Electrocardiogram. *Advances in Biometrics: Third International Conference, ICB 2009.* Italy, 2019. P. 1270–1279.

2. Chan A.D.C., Hamdy M.M., Badre A., Badee V. Wavelet Distance Measure for Person Identification Using Electrocardiograms. *IEEE Transactions on Instrumentation and Measurement*. 2008. Vol. 57. P. 248–253.

3. Safie S.I., Soraghan J.J., Petropoulakis L. Electrocardiogram (ECG) Biometric Authentication Using Pulse Active Ratio (PAR). *IEEE Transactions on Information Forensics and Security*. 2011. Vol. 6, N 4. P. 1315–1322.

4. Fainzilberg L.S., Potapova T.P. Computer Analysis and Recognition of Cognitive Phase Space Electrocardiographic Image. *Proc. of 6th International Conference on Computer analysis of Images and Patterns (CAIP-95).* Prague, 1995. P. 668–673.
5. Fang Shih-Chin, Chan Hsiao-Lung QRS detection-free electrocardiogram biometrics in the reconstructed phase space. *Pattern Recognition Letters*. 2013. Vol. 34. P. 595–602.
6. Вишневский В.В., Романенко Т.Н., Кизуб Л.А. Биометрическая идентификация человека по его электрокардиограмме. *Математичні машини і системи*. 2018. № 2. С. 88–95.
7. Вишневський В.В., Романенко Т.М., Кізуб Л.А. Біометрична ідентифікація за допомогою електрокардіограми. *Інформаційні технології та комп'ютерна інженерія*. *ITKI 2015*: П'ята міжнар. наук.-практ. конф. Івано-Франківськ, 2015. С. 130–131.
8. Vishnevsky V., Romanenko T., Kizub L. Experimental verification of possibility of human identification by the electrocardi-ogram. *5th International Conference on Application of Information and Communication Technology and Statistics and Economy and Education (ICAICTSEE – 2015).* Sofia, Bulgaria, 2015. P. 318.
9. Вишневський В.В., Романенко Т.М., Кізуб Л.А. Використання електрокардіограм і їх характеристик для ідентифікації особи. *Вісник Вінницького політехнічного інституту*. 2016. № 5. С. 7–10.
10. Вишневський В.В., Романенко Т.М., Луговський Ю.О. Валідність автентифікації людини за електрокардіограмою з обмеженою кількістю каналів. *Математичні машини і системи*. 2020. № 2. С. 43–50.
11. Вишневський В.В., Романенко Т.М. Застосування метрики Хаусдорфа для визначення нетипових кардіоциклів у тривимірному фазовому просторі координат вектор-кардіограми. *Медична інформатика і інженерія*. 2019. № 3. С. 31–36.
12. Вишневський В.В. Спосіб автоматичної автентифікації людини за її електрокардіограмою: пат. України на винахід № 117713; заявл. 15.02.17; опубл. 10.09.18, Бюл. № 17. 3.66 с.
13. URL: https://kubernetes.io/docs/concepts/overview/.

Стаття надійшла до редакції 07.06.2022