

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

СМІЛЯНЕЦЬ ФЕДІР АНДРІЙОВИЧ

УДК 004.42:519.237

ДИСЕРТАЦІЯ

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ПРИШВИДШЕННЯ ДОНАВЧАННЯ
КЛАСИФІКАТОРА ДЛЯ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ ЗА ЗОБРАЖЕННЯМИ

121 Інженерія програмного забезпечення

12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

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

_____ Ф.А. Смілянець

Науковий керівник: Фіногенов Олексій Дмитрович, к. т. н., доцент

Київ – 2025

АНОТАЦІЯ

Смілянець Ф.А. Методи та програмні засоби пришвидшення донавчання класифікатора для діагностики захворювань за зображеннями. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 – Інженерія програмного забезпечення з галузі знань 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2025.

Дисертаційна робота присвячена розробці методу класифікації медичних зображень, що спрямований на уникнення зміни топології згорткової нейронної мережі при додаванні нових класів, дозволяючи знизити накладні витрати часу на їх підтримку, та методу організації обчислень за допомогою потоків робіт, який будує граф потоку під час його виконання, чим дозволяє мінімізує час на впровадження змін у систему.

Вчасна розробка та доступність засобів тестування на інфекційні захворювання є фундаментальним компонентом контролю над епідеміями та пандеміями на кожному етапі поширення хвороби. Доступна та швидка діагностика є ключовою для своєчасності виявлення спалахів інфекції, і, як наслідок, визначення переліку та ізоляції контактних осіб чи запровадження карантинних обмежень на певній території. Упереджувальна розробка інструментів виявлення захворювань, а також пошук підходів для прискорення їх отримання та впровадження є важливим для контролю над епідеміями в майбутньому.

Одним зі способів виконання діагностики є аналіз медичних зображень за допомогою згорткових нейронних мереж. Однак, така класифікація призводить до необхідності у зміні топології моделі при виникненні потреби у додаванні нових класів. Це обмеження можна подолати через використання проміжних даних – вкладених представлень, що генеруватимуться нейронною мережею, та виконання класифікації за ними. Таким чином, нейронна мережа може бути дотренована для підтримки нового класу без зміни її топології. Алгоритми машинного навчання, що

виконують класифікацію можуть бути перетреновані для підтримки більшої кількості класів без втручання у код та за відносно короткий час.

У той час як проектування та тренування нейронних мереж широко висвітлюється в науковій літературі щодо виконання діагностики за допомогою аналізу зображень, питання практичної інтеграції моделей у програмне забезпечення практично не розглядається.

Важливим аспектом засобу для діагностики є швидкість адаптації системи до появи нових захворювань. Відтак, є потреба у зменшенні зусиль, необхідних для додавання чи модифікацію методу діагностики, який виконується у засобі. Оскільки потоки робіт дозволяють інкапсулювати окремі етапи обробки даних та гнучко змінювати їх послідовність без втручання в код системи, їх використання значно зменшує час та складність впровадження змін.

Потоки робіт є впорядкованими структурованими представленнями багатокрокових обчислювальних задач. Типовою моделлю для них є спрямовані ациклічні графи, вершини яких є певними операціями над даними, а ребра – перетікання виходів одних операцій у входи інших. Такий граф, що складається з вхідних даних, та їх послідовних перетворень за допомогою окремих вершин (також – кроків або інструментів), також часто називається пайплайном (англ. pipeline).

Існуючі системи керування потоками робіт вимагають статичного визначення графу потоку робіт розробником системи. Існування даних статичних визначень уповільнює внесення змін. Одним зі способів вирішення даної проблеми є динамічне створення та обчислення потоків робіт виходячи з наданих даних та інструментів для їх обробки і перетворення, що дозволить розширяти існуючі засоби класифікації захворювань без втручання або мінімальним втручанням людини (фахівця).

Метою дисертаційної роботи є зменшення часу адаптації програмного забезпечення аналізу медичних зображень для діагностики на основі алгоритмів машинного навчання.

У першому розділі дисертаційної роботи проведено аналіз засобів та методів діагностики захворювань за допомогою класифікації знімків комп'ютерної томографії, та розглянуто існуючі системи керування потоками робіт. За

результатами аналізу літературних джерел виділено найбільш результативні архітектури нейронних мереж та окреслено проблеми, що пов'язані з аналізом результатів. Розглянуто способи інтеграції нейронних мереж у програмне забезпечення та обґрунтовано використання потоків робіт. Проведено порівняльний аналіз сучасних систем керування потоками робіт.

У другому розділі запропоновано можливість використання вкладених представлень для вирішення задачі класифікації зображень КТ. Запропоновано модифікацію існуючої нейронної мережі для побудови вкладених представлень та доведено можливість додавання нових класів без істотної втрати точності. Розроблено математичну модель для обрахунку часу на побудову класифікатора та наведено умови, що визначають його ефективність.

У третьому розділі запропоновано метод для динамічної побудови графу потоку робіт під час його виконання, який дозволяє уникнути необхідності у його статичному визначенні вручну. Для запропонованого методу визначено основні сутності. Розроблено модель обрахунку часу виконання потоку робіт. Імітаційним моделюванням показано, що часом роботи алгоритму можна знехтувати відносно часу на корисні обчислення. Розроблено систему керування потоками робіт, що реалізує запропонований метод з використанням системи Kubernetes для розподілених кластерних обчислень та можливістю горизонтального масштабування. Проведено аналіз швидкодії системи та доведено її практичну застосовність. Розроблено математичну модель оцінки часу на побудову потоків робіт за запропонованим методом та наведено умови, що визначають межі ефективності його використання.

У четвертому розділі виконано проектування та реалізацію програмного забезпечення аналізу зображень КТ для діагностики захворювань. Представлено архітектуру програмного забезпечення. Розроблено низку обчислюючих засобів системи керування потоками робіт для виконання класифікації зображень комп'ютерної томографії, у тому числі за допомогою вкладених представлень. Реалізовано сервер та інтерфейс користувача системи. Наведено опис функціональності системи.

- Результати, отримані у дисертаційному дослідженні, містять наукову новизну
- **вперше запропоновано** метод динамічного конструювання потоків робіт під час їх виконання, який відрізняється від наявних автоматичною побудовою графу виконання, що дозволяє виключити етап визначення статичного потоку робіт та зменшити час на впровадження змін у програмне забезпечення;
 - **удосконалено** метод класифікації зображень шляхом використання вкладених представлень, що дозволяє додавання нових класів без зміни топології нейронної мережі;
 - **удосконалено** математичну модель для оцінки часу адаптації програмного забезпечення класифікації зображень шляхом врахування складових часу модифікації класифікатора та побудови графу потоку робіт, що дає можливість порівнювати швидкості впровадження змін у програмне забезпечення та здійснювати обґрунтований вибір методів та архітектурних рішень.

Розроблено хмарне програмне забезпечення для автоматизованої діагностики захворювань за допомогою аналізу знімків КТ. Запропоновані методи, застосовані у програмному засобі, дозволяють швидке дотренування та розширення для підтримки нових методів, розпізнавання нових захворювань тощо. Представлений метод динамічного конструювання потоків робіт та система на його основі можуть бути використані як заміна поточних систем організації обчислень у сфері біоінформатики, у тому числі для проведення геномних розрахунків.

Розроблені в дисертації програмні засоби можуть використовуватись як окремо (СКПР, ПЗ аналізу знімків КТ для діагностики) так і в складі інших систем (наприклад, як складові систем підтримки прийняття рішень).

Результати проведених досліджень було опубліковано у 9 наукових працях, з яких 4 у фахових наукових журналах категорії «Б», 1 у журналі, що індексується наукометричною базою даних Scopus, 4 у матеріалах міжнародних науково-практичних конференцій.

Ключові слова: інженерія програмного забезпечення, аналіз медичних даних, знімки комп'ютерної томографії, вкладені представлення, класифікатори, машинне навчання, нейронні мережі, згорткові нейронні мережі, потоки робіт, системи виконання потоків робіт, хмарні обчислення, розподілені обчислення, контейнер.

ABSTRACT

Smilianets F.A. Methods and software Tools for accelerating classifier additional training for image-based disease diagnosis. – Qualifying scientific work on the rights of the manuscript.

Thesis for the degree of Doctor of Philosophy in specialty 121 – Software Engineering of knowledge field 12 – Information Technologies. – National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, 2025.

The dissertation focuses on developing a medical image classification method aimed at avoiding changes in convolutional neural network topology when adding new classes, thus reducing the time overhead for their support, and a workflow-based computation organization method that constructs the pipeline graph during execution, thereby minimizing the time required for system modifications.

The timely development and availability of infectious disease testing tools is a fundamental component of epidemic and pandemic control at every stage of disease spread. Accessible and rapid diagnostics are crucial for timely detection of infection outbreaks and, consequently, for identifying and isolating contacts or implementing quarantine restrictions in specific areas. Proactive development of disease detection tools, as well as finding approaches to accelerate their acquisition and implementation, is essential for control of future epidemics.

One approach to diagnosis involves analyzing medical images using convolutional neural networks. However, such classification necessitates changes in model topology when new classes need to be added. This limitation can be overcome through the use of intermediate data – embeddings generated by the neural network – and performing classification based on them. Thus, the neural network can be fine-tuned to support new classes without introducing changes to its topology. Machine learning algorithms performing the classification on embeddings can be retrained to support more classes without code intervention and in a relatively short time.

While the design and training of neural networks for image-based diagnostics is extensively covered in scientific literature, the practical aspects of integrating these models into software systems remain largely unexplored.

An important aspect of diagnostic tools is the speed at which the system can adapt to emerging diseases. Therefore, there is a need to reduce the effort required for adding or modifying diagnostic methods implemented in the tool. Since workflows allow encapsulation of individual data processing stages and flexible alteration of their execution sequence without system code intervention, their use significantly reduces the time and complexity of implementing changes.

Workflows are ordered, structured representations of multi-step computational tasks. They are typically modeled as directed acyclic graphs, where vertices represent data operations and edges represent the flow of outputs from one operation to inputs of others. Such a graph, consisting of input data and their sequential transformations through individual vertices (also known as steps or tools), is often referred to as a pipeline.

Existing workflow management systems require static workflow graph definitions by system developers. The existence of these static definitions slows down the implementation of changes. One solution to this problem is dynamic creation and computation of workflows based on provided data and tools for their processing and transformation, which enables expansion of existing disease classification tools with minimal or no human (specialist) intervention.

The aim of the thesis is to reduce the adaptation time of machine learning based software for medical image analysis for diagnostics.

The first section analyzes tools and methods for disease diagnosis through classification of computed tomography images and examines existing workflow management systems. Based on the literature review, the most effective neural network architectures are identified and problems related to results analysis are outlined. Methods of integrating neural networks into software are considered and the use of workflows is justified. A comparative analysis of modern workflow management systems is conducted.

The second section proposes the use of embeddings for solving CT image classification tasks. A modification of an existing neural network for generating embeddings is proposed and the possibility of adding new classes without significant accuracy loss is demonstrated. A mathematical model for calculating classifier construction time is developed and conditions determining its efficiency are presented.

In the third section, a method for the dynamic construction of a workflow graph during its execution is proposed, which allows avoiding the need for its static manual definition. The main entities for the proposed algorithm are defined. A model for calculating the execution time of the workflow is developed. Through simulation modeling, it is demonstrated that the algorithm's runtime overhead is negligible compared to useful computation time. A workflow management system that implements the proposed algorithm using the Kubernetes system for distributed cluster computing with the ability to scale horizontally has been developed. The performance of the system is analyzed, and its practical applicability is proven. A mathematical model for estimating workflow construction time using the proposed method is developed, and conditions determining its effectiveness boundaries are presented.

In the fourth section, the design and implementation of a CT image analysis software for disease diagnosis. The architecture of the software is presented. A number of computing tools have been developed in the workflow management system for performing classification of computed tomography images, including using embedded representations. The server and user interface of the software have been implemented. The functionality of the software is described.

The results obtained in the dissertation research contain scientific novelty:

- **for the first time**, a method for the dynamic construction of workflows during their execution has been proposed, which is characterized by the automatic construction of the execution graph and allows to avoid the stage of manually defining a static workflow, and thus reduces the time for implementing changes in software;
- an image analysis method was **improved** by using embeddings, which enables the addition of new classes without modifying neural network topology;
- mathematical model for estimating image classification software adaptation time was **improved** by incorporating classifier modification and workflow graph construction time components, enabling comparison of software modification implementation speeds and facilitating informed selection of methods and architectural solutions.

A cloud-based software tool has been developed for automated disease diagnosis through the analysis of CT scans. The proposed methods, applied in the software tool, allow rapid fine-tuning and expansion to support new methods, recognition of new diseases, etc. The presented method of dynamic workflow formation and the system based on it can be used as a replacement for the current systems of organizing computations in the field of bioinformatics, including for performing genomic calculations.

The software tools developed in the dissertation can be used both independently (Workflow Management System, CT scan analysis software for diagnostics) and as components of other systems (for example, as elements of decision support systems).

The results of the research conducted were published in 9 scientific papers, of which 4 were in professional scientific journals, 1 was in a journal indexed by Scopus, and 4 were in the materials of international scientific and practical conferences.

Keywords: software engineering, medical data analysis, computed tomography scans, embeddings, classifiers, machine learning, neural networks, convolutional neural networks, workflows, workflow management systems, cloud computing, distributed computing, containers.

Список публікацій здобувача

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. **F. Smilianets** and O. Finogenov, “Multi-Class Classification of Pulmonary Diseases Using Computer Tomography Images,” *Adaptive Systems of Automatic Control*, vol. 2, no. 43, pp. 78–83, Dec. 2023, doi: 10.20535/1560-8956.43.2023.292255.
2. **F. Smilianets**, “Application of Transfer Learning for enhanced pulmonary disease detection via CT image embeddings,” *Adaptive Systems of Automatic Control*, vol. 1, no. 44, pp. 24–29, 2024, doi: 10.20535/1560-8956.44.2024.302198.
3. **F. Smilianets** and O. Finogenov, “Running a workflow without workflows: a basic algorithm for dynamically constructing and traversing an implied directed acyclic graph in a non-deterministic environment,” *Informatyka, Automatyka, Pomiarы w Gospodarce i Ochronie Środowiska*, vol. 14, no. 1, pp. 115–118, Mar. 2024, doi: 10.35784/iargos.5858 (проіндексовано в **Scopus**).
4. **F. A. Smilianets** and O. D. Finogenov, “Review of disease identification methods based on computed tomography imagery,” *Ukrainian Journal of Information Technology*, vol. 6, no. 1, pp. 95–100, 2024, doi: 10.23939/ujit2024.01.095.
5. **F. Smilianets**, “Application of embeddings for multi-class classification with optional extendability,” *Adaptive Systems of Automatic Control*, vol. 2, no. 45, pp. 186–193, Oct. 2024, doi: 10.20535/1560-8956.45.2024.313198

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. **Смілянець Ф.А.**, Фіногенов О.Д. Огляд методів ідентифікації захворювань на основі знімків комп’ютерної томографії. XII Міжнародна науково-практична конференція «Комплексне забезпечення якості технологічних процесів та систем», 2022, ISBN 978-617-7932-34-4.
7. **Смілянець, Ф. А.**, Фіногенов О.Д. Мультикласова класифікація легеневих захворювань за допомогою знімків комп’ютерної томографії. Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023): матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів, присвячених 125-й річниці КПІ ім. Ігоря Сікорського (9-11 травня 2023 р., Київ). – Київ : КПІ ім. Ігоря Сікорського, ІПІ ФІОТ, 2023. с. 28-30.

8. **Смілянець, Ф. А.,** Фіногенов О.Д. Методи та програмне забезпечення обробки знімків комп'ютерної томографії для автоматизації діагностики захворювань. Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2024): матеріали VI Міжнародної науково-практичної конференції молодих вчених та студентів, 21-23 травня 2024 року – Київ : КПІ ім. Ігоря Сікорського, ІІІ ФІОТ, 2024. с. 124-127.
9. **Смілянець, Ф. А.,** Фіногенов О.Д. Порівняння алгоритмів класифікації з використанням вкладених представлень знімків КТ. Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2024): матеріали VII Міжнародної науково-практичної конференції молодих вчених та студентів, 20-22 листопада 2024 року – Київ : КПІ ім. Ігоря Сікорського, ІІІ ФІОТ, 2024. с. 96-100.

ЗМІСТ

| | |
|--|-----------|
| ЗМІСТ | 13 |
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ | 15 |
| ВСТУП | 16 |
| 1 МЕТОДИ ТА ЗАСОБИ ВИКОНАННЯ АНАЛІЗУ ЗОБРАЖЕНЬ ДЛЯ | |
| ДІАГНОСТИКИ ЗА ДОПОМОГОЮ ГЛИБОКОГО НАВЧАННЯ..... | 21 |
| 1.1 Глибоке навчання та аналіз зображень для діагностики..... | 21 |
| 1.2 Методи класифікації знімків КТ | 22 |
| 1.2.1 Проблеми, що пов'язані з аналізом результатів..... | 30 |
| 1.3 Використання потоків робіт у аналізі зображень для діагностики..... | 32 |
| 1.4 Засоби для керування потоками робіт..... | 36 |
| 1.5 Висновки до розділу..... | 41 |
| 2 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ ДЛЯ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ | 43 |
| 2.1 Формування набору даних | 43 |
| 2.2 Мультикласова класифікація знімків КТ..... | 45 |
| 2.3 Використання вкладених представлень для класифікації зображень | 48 |
| 2.3.1 Побудова вкладених представлень зображень та виконання | |
| класифікації за ними | 49 |
| 2.3.2 Аналіз ефективності класифікаторів на основі | |
| вкладених представлень | 54 |
| 2.3.3 Аналіз часових характеристик використання вкладених | |
| представлень для класифікації | 62 |
| 2.4 Мультикласова класифікація знімків КТ з використанням | |
| вкладених представлень | 65 |
| 2.4.1 Побудова вкладених представлень знімків КТ | 65 |
| 2.4.2 Застосування алгоритмів машинного навчання до класифікації | |
| вкладених представлень знімків КТ..... | 67 |
| 2.5 Висновки до розділу..... | 70 |
| 3 ПРОГРАМНА СИСТЕМА КЕРУВАННЯ ПОТОКАМИ РОБІТ | 73 |

| | | |
|-------|---|------------|
| 3.1 | Динамічна побудова графу потоку робіт під час його виконання | 74 |
| 3.1.1 | Складові частини потоку робіт | 75 |
| 3.1.2 | Опис реалізації методу | 75 |
| 3.2 | Програмна система керування потоками робіт з динамічною побудовою графу | 83 |
| 3.2.1 | Компоненти системи | 84 |
| 3.2.2 | Програмна реалізація системи | 87 |
| 3.2.3 | Програмний інтерфейс взаємодії з СКПР | 91 |
| 3.3 | Аналіз швидкодії СКПР | 98 |
| 3.3.1 | Аналіз швидкодії алгоритму в ізоляції | 100 |
| 3.3.2 | Аналіз швидкодії СКПР у комплексі | 105 |
| 3.4 | Аналіз часових характеристик використання систем керування потоками робіт з динамічною побудовою графу під час виконання | 110 |
| 3.5 | Висновки до розділу | 112 |
| 4 | ПРОГРАМНИЙ ЗАСІБ АНАЛІЗУ ЗНІМКІВ КТ ДЛЯ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ | 114 |
| 4.1 | Компоненти програмного засобу | 116 |
| 4.2 | Реалізація | 122 |
| 4.2.1 | Оператори для підсистеми керування потоками робіт | 122 |
| 4.2.2 | Серверна частина програмного засобу | 125 |
| 4.2.3 | Клієнтська частина програмного засобу | 128 |
| 4.3 | Пошук найближчих за допомогою векторної бази даних | 129 |
| 4.4 | Висновки до розділу | 133 |
| | ВИСНОВКИ | 134 |
| | СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ | 136 |
| | ДОДАТОК А. МАТРИЦІ НЕВІДПОВІДНОСТІ КЛАСИФІКАТОРІВ, НАВЧЕНИХ НА ВКЛАДЕНИХ ПРЕДСТАВЛЕННЯХ ЗНІМКІВ КТ | 148 |
| | ДОДАТОК Б. ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОГРАМНИМ ЗАСОБОМ ... | 152 |
| | ДОДАТОК В. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА | 156 |

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | |
|-----------|--|
| КТ | – Комп'ютерна томографія |
| ШНМ, НМ | – Штучна нейронна мережа |
| ЗШНМ, ЗНМ | – Згорткова штучна нейронна мережа |
| ПР | – Потік робіт |
| СКПР | – Система керування потоками робіт |
| ВП | – Вкладене представлення (англ. embedding) |
| ВД | – Вид даних |
| ОД | – Одиниця даних |
| СКБД | – Система керування базами даних |
| Датасет | – Набір даних |

ВСТУП

Актуальність теми. Інфекційні хвороби супроводжують людство з доісторичних часів. Поява сільського господарства з одомашненням тварин та вирощуванням рослин близько 10,000 років тому дозволила більші та компактніші поселення людей, а також дало більше можливостей для передачі патогенів між видами (зоонозність). Ці та інші фактори стали ключовими у появі епідемій як явища [1].

Глобалізація та торгово-економічна інтеграція сприяли поглибленню контактів між географічно віддаленими регіонами та спільнотами, а відтак – інтенсивному обміну патогенами між ними. Так, у класичну епоху обмін патогенами та їх носіями відбувався між великими євразійськими цивілізаціями – середземноморськими державами, Близьким Сходом, Індією та Китаєм, а епоха ренесансу містить багато випадків передачі патогенів корінним жителям Америки та островів Тихого океану [1].

Подальша інтенсифікація глобалізації, та поява авіаційного сполучення значно підвищили швидкість поширення інфекційних захворювань. Так, з 12 найважчих епідемій в історії людства, 5 відбулись у 20-му столітті, та 2 – у 21-му, причому останні 4 відбулись з інтервалом у 15 років [2].

Вчасна розробка та доступність діагностичних тестів є фундаментальним компонентом контролю над епідеміями та пандеміями на кожному етапі поширення хвороби [3]. Доступна та швидка діагностика є ключовим компонентом своєчасності виявлення спалахів інфекції, і, як наслідок, визначення переліку та ізоляції контактних осіб чи запровадження карантинних обмежень на певній території.

Недостатня готовність діагностичних інструментів зіграла істотну роль у затримці реакції на ряд епідемій, включаючи хвороби, спричинені вірусами Ебола [4] та Зіка [5], тощо.

Аналогічно, на початку пандемії COVID-19 тестування за допомогою методу ПЛР було дорогим, займало багато часу (декілька днів) та кількість тестів була обмеженою. Тестування на COVID-19 стало доступним лише зі здешевленням ПЛР

та поширенням інших методів тестування (таких, як антиген-тестування). До цього, найшвидшим способом підтвердження діагнозу COVID-19 був аналіз знімка комп'ютерної томограми лікарем-радіологом на предмет характерних для вірусу уражень легень.

Під час появи та поширення наступної інфекції не буде достатньо часу на розробку діагностичних засобів з повним дотриманням усіх регуляторних правил на спеціалізованих до патогену платформах. Доступ до діагностичних засобів є проблемою навіть для наявних інфекцій – так, для 20 патогенів, що мають потенціал до спричинення спалахів, епідемій, та пандемій, готовність систем для тестування є критично недостатньою [6].

Відтак, упереджувальна розробка діагностичних інструментів, а також пошук підходів для швидшого отримання засобів діагностики є важливими для контролю над епідеміями в майбутньому [3], [4].

Внаслідок пандемії COVID-19, існують великі загальнодоступні анотовані набори даних зі знімками КТ, що є ідеальними для застосування засобів машинного навчання з метою проведення аналізу знімків комп'ютерної томографії для діагностики.

Комп'ютерна томографія (КТ) є методом рентгенівського сканування, що використовується у радіології для неінвазивного отримання високоякісних зображень тіла людини для діагностичних цілей.

Знімки методом комп'ютерної томографії отримуються за допомогою комп'ютерних алгоритмів формування зображення з інформації отриманої датчиками під час поступового та пошарового проходу пучка рентгенівських променів через тіло людини під різними кутами.

Результатом є тривимірне пошарове зображення внутрішніх органів людини, де товщина шару може варіюватися від 1.5 мм до 10 мм, а детальність шару залежить від потужності пучка рентгенівських променів під час виконання знімка. Ці змінні залежать від об'єкта діагностики та характеристик пацієнта. Наприклад, за наявності такої можливості, а особливо для вагітних жінок і дітей, перевагу надають низькодозовим КТ-знімкам.

Знімки КТ є одним з основних методів діагностики захворювань легень, оскільки дає тривимірне та детальне зображення, процес зняття якого може бути підлаштований для процесу діагностики конкретного захворювання, а також дає легку можливість для побудови довільних зрізів простору, відзнятого знімком КТ.

Відтак, знімки КТ є ідеальним простором для розробки нових підходів, методів та засобів комп'ютерної діагностики. Розроблювані підходи та методи мусять бути спрямовані на уможливлення їх якнайшвидшого оновлення та адаптації для протидії спалахам як відомих, так і невідомих хвороб у локальних та глобальних масштабах.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота частково пов'язана з науковими розробками, що здійснюються на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» за науковим напрямом “Методи та технології високопродуктивних обчислень та обробки надвеликих масивів даних”. Державний реєстраційний номер 0117U000924.

Мета і завдання дослідження. Зменшення часу адаптації програмного забезпечення аналізу медичних зображень для діагностики на основі алгоритмів машинного навчання.

Для досягнення вказаної мети було сформовано наступні завдання:

1. дослідити методи аналізу знімків КТ;
2. розробити метод класифікації знімків КТ, що дозволяє зменшити час донавчання класифікатора при додаванні нових класів;
3. розробити метод організації розподілених обчислень, що дозволяє зменшити час на адаптацію програмного забезпечення при впровадженні змін;
4. розробити програмне забезпечення аналізу знімків КТ для діагностики захворювань з можливістю використання декількома організаціями.

Об'єкт дослідження: процес аналізу зображень для діагностики захворювань.

Предмет дослідження: методи організації класифікації зображень на основі нейронних мереж.

Методи дослідження. методи машинного навчання, методи класифікації, згорткові нейронні мережі, програмні засоби кластерних обчислень, методи об'єктно-

орієнтованого та процедурного програмування, методи та підходи до розподілених обчислень.

Наукова новизна отриманих результатів:

- *вперше запропоновано* метод динамічного конструювання потоків робіт під час їх виконання, який відрізняється від наявних автоматичною побудовою графу виконання, що дозволяє виключити етап визначення статичного потоку робіт та зменшити час на впровадження змін у програмне забезпечення;
- *удосконалено* метод класифікації зображень шляхом використання вкладених представлень, що дозволяє додавання нових класів без зміни топології нейронної мережі;
- *удосконалено* математичну модель для оцінки часу адаптації програмного забезпечення класифікації зображень шляхом врахування складових часу модифікації класифікатора та побудови графу потоку робіт, що дає можливість порівнювати швидкості впровадження змін у програмне забезпечення та здійснювати обґрунтований вибір методів та архітектурних рішень.

Практичне значення отриманих результатів. Розроблено хмарний програмний засіб аналізу знімків КТ для діагностики захворювань. Запропоновані методи, застосовані у програмному засобі, дозволяють швидке дотренування та розширення для підтримки нових методів, розпізнавання нових захворювань тощо. Запропонований метод динамічного конструювання потоків робіт може бути використаний в системах організації обчислень у сфері біоінформатики, у тому числі для проведення геномних розрахунків.

Особистий внесок здобувача. Дисертація та усі результати представлені у ній отримані автором особисто у процесі самостійних наукових досліджень. У роботах, написаних у співавторстві, дисертантові належать наступні результати:

- у [7], [8] реалізація мультикласового класифікатора, його експериментальне дослідження;

- у [9] створення генератора вкладених представлень та мультикласових класифікаторів, дослідження втрати точності при додаванні нового класу;
- у [10], [11] створення генератора вкладених представлень зображень КТ, мультикласового класифікатора, його експериментальне дослідження;
- у [12] проведення обчислювальних експериментів та порівняння алгоритмів класифікації з використанням вкладених представлень;
- у [13], [11] розробка, обґрунтування, реалізація та експериментальне дослідження алгоритму динамічного конструювання графів потоків робіт;
- у [14], [15] збір та аналіз інформації про існуючі дослідження.

Апробація результатів дисертації. Результати роботи було опубліковано та обговорено на всеукраїнських та міжнародних конференціях: XII Міжнародна науково-практична конференція «Комплексне забезпечення якості технологічних процесів та систем», 2022; IV Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)» присвячена 125-й річниці КПІ ім. Ігоря Сікорського; VI Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (Soft Tech-2024), VII Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (Soft Tech-2024).

Публікації. Результати проведених досліджень було опубліковано у 9 наукових працях, з яких 4 у фахових наукових журналах категорії «Б», 1 у журналі, що індексується наукометричною базою даних Scopus, 4 у матеріалах міжнародних науково-практичних конференцій.

Структура і обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 113 найменувань та додатків. Загальний обсяг дисертації становить 157 сторінок, з яких 147 сторінок основного тексту, 3 додатки на 10 сторінках, та містить 31 рисунок, 43 формули, 15 таблиць.

1 МЕТОДИ ТА ЗАСОБИ ВИКОНАННЯ АНАЛІЗУ ЗОБРАЖЕНЬ ДЛЯ ДІАГНОСТИКИ ЗА ДОПОМОГОЮ ГЛИБОКОГО НАВЧАННЯ

1.1 Глибоке навчання та аналіз зображень для діагностики

Глибоке навчання є підмножиною методів машинного навчання, яке стосується навчання штучних нейронних мереж (ШНМ) – математичних систем, виконаних за подобою нейронних структур мозку тварин та людини. ШНМ складаються з пов'язаних між собою нейронів. Основним принципом роботи ШНМ є послідовне перетворення інформації паралельними ланцюгами нейронів, що передають інформацію між собою та до наступних нейронів у мережі. Як правило, нейрони мереж організовані у шари – послідовно поєднаних наборів нейронів, від вхідного до вихідного шару.

ШНМ застосовуються з метою виділення ознак у даних та застосування цих ознак для виконання певної задачі, часто – класифікації, різних форм кодування та кластеризації тощо.

У сфері медичних досліджень ШНМ застосовуються для пошуку нових препаратів, аналізу медичних зображень, автоматичної діагностики тощо. Одним з найбільш частих застосувань глибокого навчання в медичній сфері є аналіз даних отриманих з медичних досліджень у процесі діагностики – кардіограм (стандартних та добових), енцефалограм, генетичних даних отриманих за допомогою мікромасивів [16], аналізі метаболітів у біологічних рідинах [16], рентгенографічних знімків, знімків магнітно-резонансної томографії та рентгенівської комп'ютерної томографії.

Основним класом глибоких нейронних мереж, що застосовуються для аналізу знімків комп'ютерної томографії є згорткові штучні нейронні мережі (ЗШНМ).

Згорткові штучні нейронні мережі, вперше представлені у [17] є класом глибоких штучних нейронних мереж, винайденим з метою мінімізації попередньої обробки вхідних даних та розв'язання проблеми експоненційного росту кількості параметрів нейронної мережі при збільшенні розмірності вхідних даних. На прикладі аналізу зображень, в нейронній мережі побудованій на основі перцептронів, кожен

нейрон у вхідному шарі нейронної мережі мусить мати тисячі параметрів для аналізу зображення розміром 100 на 100 пікселів. Натомість, ЗШНМ здійснюють пошук та вивчення ознак застосовуючи послідовні згорткові операції над вхідними даними, що дозволяє знизити кількість необхідних нейронів, і, відповідно, об'єм необхідних обчислень на порядки. Згортка, у випадку нейронних мереж, виконується послідовним обчисленням скалярного добутку між певним вікном у вхідних даних шару та «ядром» нейрону, який є маскою (фільтром) для пошуку тих чи інших ознак. На кожному кроці обчислення, вікно у вхідних даних зміщується та обчислюється наступний скалярний добуток. В результаті, нейрон продукує матрицю отриманих скалярних добутків. Ці скалярні добутки і є віднайденими шаром ознаками. Таким чином, послідовною обробкою даних декількома шарами нейронна мережа має здатність визначати та ідентифікувати більш абстрактні ознаки.

1.2 Методи класифікації знімків КТ

Через обмеження, пов'язані з наявними технологіями та доступними обчислювальними ресурсами, ранні роботи з аналізу знімків КТ для діагностики вимагали інтенсивної попередньої обробки даних та подекуди фокусувались на визначенні специфічних, аніж загальних ознак.

Так, у [18] було побудовано декілька багатоступневих класифікаторів, орієнтованих на пошук широкого фіброзу легень – рубцювання тканин легень, що призводить до проблем з диханням та доступом кисню у кров. Для визначення діагнозу, DICOM-зображення сегментувались та дооброблялись для відділення тканин легень від простору, який відображає заповнені повітрям легені. Отримані сегменти аналізувались на наявність ознак за текстурними та іншими атрибутами. На основі отриманих ознак проводилась класифікація за допомогою ряду правил, визначених за допомогою машинного навчання. Схожим чином, за допомогою пошуку ознак статистичними методами та їх аналізу за допомогою класифікатора на основі методу опорних векторів, у [19] було побудовано систему для визначення характеру ураження печінки – відсутність ураження, гемангіома, гепатоцелюлярна

карцинома. Однак, на відміну від попереднього дослідження, у [19] розглядалась класифікація окремих регіонів КТ-знімка, виділених для цього вручну досвідченим лікарем-радіологом.

Натомість, [20] розглядає буквальний підхід до пошуку та ідентифікації різних видів травм голови. В цьому дослідженні КТ-зображення голови людини сегментувались з метою пошуку потенційних областей внутрішньої кровотечі за допомогою базових математичних технік. Характеристики віднайдених регіонів, такі як площа, довжина, розміщення регіону ураження оброблялись за допомогою дерева прийняття рішень.

З розвитком методів глибокого навчання та різким стрибком у доступності високопродуктивних комп'ютерів, акцент у розробці систем обробки зображень загалом та медичних зображень зокрема змістився з розробки специфічних систем до застосування одноманітних узагальнених методів, як правило заснованих на згорткових нейронних мережах. За результатами дослідження [14] майже усі статті, спрямовані на аналіз КТ-знімків легень були засновані на різних архітектурах згорткових нейронних мереж.

З відкритих джерел було відібрано 39 наукових робіт на тему діагностики на основі машинного аналізу знімків КТ, опублікованих між січнем 2017 року та вереснем 2023 року. Для кожної з них було зібрано цільове захворювання, використаний метод, розмір набору даних, методи оцінювання досягнутого результату та класи, що наявні у вхідному наборі даних. Аналізувались наступні параметри:

- захворювання, виявлення якого є метою;
- архітектура нейронної мережі;
- розміри датасету;
- використані метрики та отримані значення;
- класи зображень, які може розрізнити натренована мережа.

Опубліковані роботи було відсортовано у дві категорії:

- до початку пандемії COVID-19;
- після початку пандемії COVID-19.

Повний список знайдених публікацій наведено у Таблицях 1.1 та 1.2, наведених у додатку X.

Таблиця 1.1 – Дослідження, опубліковані до пандемії COVID-19

| Рік Цит. | Цільове захв. | Архітектура | Розмір датасету | Використані метрики | Класи датасету |
|-----------|---------------|----------------------|-----------------|-------------------------------|--|
| 2017 [21] | P.T. | AlexNet GoogLeNet | 1007 | AUC=99 % | Туберкульоз легень, здорові легені |
| 2017 [22] | L.C. | Unknown | 48 | ACC=64.6 % AUC=64.6 % | Не вказано |
| 2017 [23] | Sg. | CNN | 240 | ACC=88 % | Знімки КТ |
| 2018 [24] | L.N. | Modified ResNet50 | Unknown | ACC=91.6% AUC=95.7 % | Доброякісні/злюякісні легеневі вузли |
| 2018 [25] | L.N. | Method comparison | 43 292 | AUC=92-99 % | Наявність та відсутність вузла |
| 2018 [26] | L.N. | DetectNet YOLO | 1018 | ACC=93 % | Наявність та відсутність вузла |
| 2019 [27] | Pt. | CNN | 80 | Sensit.=100 % Specif.=83 % | Пневмоторакс, здорові легені |
| 2019 [28] | L.C. | Unknown | 1139 | ACC=94.4 % | Рак легень, здорові легені |
| 2019 [29] | Pt. | CNN YOLO | 1596 | ACC=86-87 % | Різні стадії пневмотораксу |
| 2019 [30] | L.N. | CMixNet | 888 | Sensit.=94 % Specif.=91 % | Доброякісні/злюякісні легеневі вузли |
| 2019 [31] | Em. | CNN LSTM | 7143 | Unknown | Різні типи емфіземи |
| 2019 [32] | L.C. | 3DDCNN | 55 | ACC=98.51 % | Доброякісні/злюякісні легеневі вузли, здорові легені |

Таблиця 1.2 – Дослідження, опубліковані після початку пандемії COVID-19

| Рік Цит. | Цільове захв. | Метод | Розмір датасету | Використані метрики | Класи датасету |
|-----------|---------------|--|-----------------|---|---|
| 2020 [33] | CVD | UNet, WeakLabel | 499 | ACC=90 % AUC=95.9 % | Наявність та відсутність COVID-19 |
| 2020 [34] | CVD | ResNet-50, Xception, Inception-v3, VGG16 | 3993 | Sensit.=99 % Specif.=100 % ACC=99.8 % | COVID-19, інші пневмонії, інші захворювання |
| 2020 [35] | CVD | DenseNet, ResNet | 812 | F1=90 %, AUC=98 %, ACC=89 % | Наявність та відсутність COVID-19 |

| Рік Цит. | Цільове захв. | Метод | Розмір датасету | Використані метрики | Класи датасету |
|-----------|---------------|--|---------------------------------|--|--|
| 2020 [36] | CVD | VGG-16, ResNet, DenseNet, EfficientNet, CRNet. | 349 | F1=85 % AUC =94 % | COVID-19, інші захворювання, здорові легені |
| 2020 [37] | CVD | 2D ROI, GradCam | 110 | AUC=94.8 % | Наявність та відсутність COVID-19 |
| 2020 [38] | CVD | Unknown | 14,435 | F1=97 % | COVID-19, інші пневмонії |
| 2020 [39] | L.C. | AlexNet | Невідомо | ACC=97.2 % | Не вказано |
| 2020 [40] | CVD | ImageNet | 610 пацієнтів | ACC=98.5 % | COVID-19, інші пневмонії, туберкульоз легень |
| 2020 [41] | CVD | ResNet50 | 6868 зображень 400 пацієнтів | ACC=95.6 % | COVID-19, здорові легені |
| 2020 [42] | CVD | DenseNet3D, MNas3DNet ResNet3D | 3,993 | ACC=87.1 %, F1=87.25 %, AUC=95.7 % | COVID-19, інші пневмонії, здорові легені |
| 2020 [43] | CVD | ResNet50 | 720 | ACC=92.2 % | Наявність та відсутність COVID-19 |
| 2020 [44] | P.T. | CNN | 1002 | Precis.=94 % Recall=98 % | Туберкульоз, здорові легені |
| 2020 [45] | CVD | EfficientNet | 544 | ACC=89.7 %, F1=89.6 %, AUC=89.5 % | COVID-19 та інші |
| 2021 [46] | CVD | COVID-Net | 13,975 | ACC=94.3 % | COVID-19, інші пневмонії, туберкульоз легень |
| 2021 [47] | CVD | EfficientNetB3 | 7184 | AUC=95 % | COVID-19, інші пневмонії, туберкульоз легень |
| 2021 [48] | CVD | CNN | 3228 | ACC=99 % | Наявні набори даних SARS-CoV-2 CT-scan та COVID19-CT |
| 2021 [49] | CVD | ResNet50 | 2592 | Sensit.=93 % Specif.=92 % | COVID-19 та інші |
| 2021 [50] | CVD | CNN | 465 | Не вказано | COVID-19 |
| 2021 [51] | CVD | VGG-16, ResNet50, Xception | SARS-CoV-2 CT | ACC=98.79 % F1 =0.99 | Набір даних SARS-CoV-2 CT |
| 2021 [42] | CVD | Differential architecture search | 3,993 | ACC=88-96 % для різних архітектур | Неуточнені «наявні набори даних» |
| 2022 [52] | CVD | VGG, ResNet | 757 | ACC=99.35 %, ACC=96.77 % | Наявність та відсутність COVID-19 |

| Рік Цит. | Цільове захв. | Метод | Розмір датасету | Використані метрики | Класи датасету |
|-----------|---------------|------------------------|-----------------|---------------------|--|
| 2022 [53] | CVD | Inception, ResNetV2 | 2,471 | ACC=96 % | Наявність та відсутність COVID-19 |
| 2022 [54] | CVD | SeNet154 | 31590 | ACC=98 % | COVID-19, інші пневмонії, здорові легені |
| 2022 [55] | CVD | P-DenseCOVNet. | 2,698 | ACC=87.5 % | COVID-19, інші пневмонії, здорові легені |
| 2022 [56] | CVD | CVD19-Net | 13216 | ACC=98 % | Наявність та відсутність COVID-19 |
| 2023 [57] | CVD | DarkNet19, MobileNetV2 | 2482 | ACC=98 % | Наявність та відсутність COVID-19 |
| 2023 [58] | CVD | RADIC | 2482 | ACC=99 % | Наявність та відсутність COVID-19 |

* де:

- L.N. – легеневі вузли;
- P.T. – туберкульоз легень;
- Pt. – пневмоторакс;
- L.C. – рак легень;
- Sg. – сегментація зображень;
- Em. – емфізема;
- CVD – COVID-19;
- ACC – точність;
- AUC – Area Under Curve, площа під ROC-кривою [59];
- Sensit. – чутливість;
- Specif. – специфічність;
- Precis. – влучність;
- Recall – повнота.

З 39 знайдених робіт, 12 були опубліковані до початку пандемії COVID-19: з них 7 належать до ідентифікації та класифікації легеневих новоутворень – вузлів та пухлин, 2 мають на меті пошук пневмотораксу, 1 туберкульозу легень та 1 емфіземи легень, та 1 має за мету сегментацію знімка. 27 з 39 робіт були опубліковані після початку пандемії COVID-19, 25 з них стосуються ідентифікації COVID-19, 1 туберкульозу та 1 раку легень. Така різка зміна фокуса досліджень демонструє, швидко реакцію наукової спільноти на нагальну потребу в автоматизованих засобах діагностики COVID-19.

Очікувано, що 2020 рік був піковим у публікаціях щодо автоматизованого розпізнавання захворювань за знімком КТ грудної клітини. На (рис. 1.1) видно, що з 2020 дана тема стійко втрачає інтерес, що можна пояснити як появою більш доступних методів тестування на COVID-19, так і загальним зниженням актуальності даної проблеми в контексті пандемії.

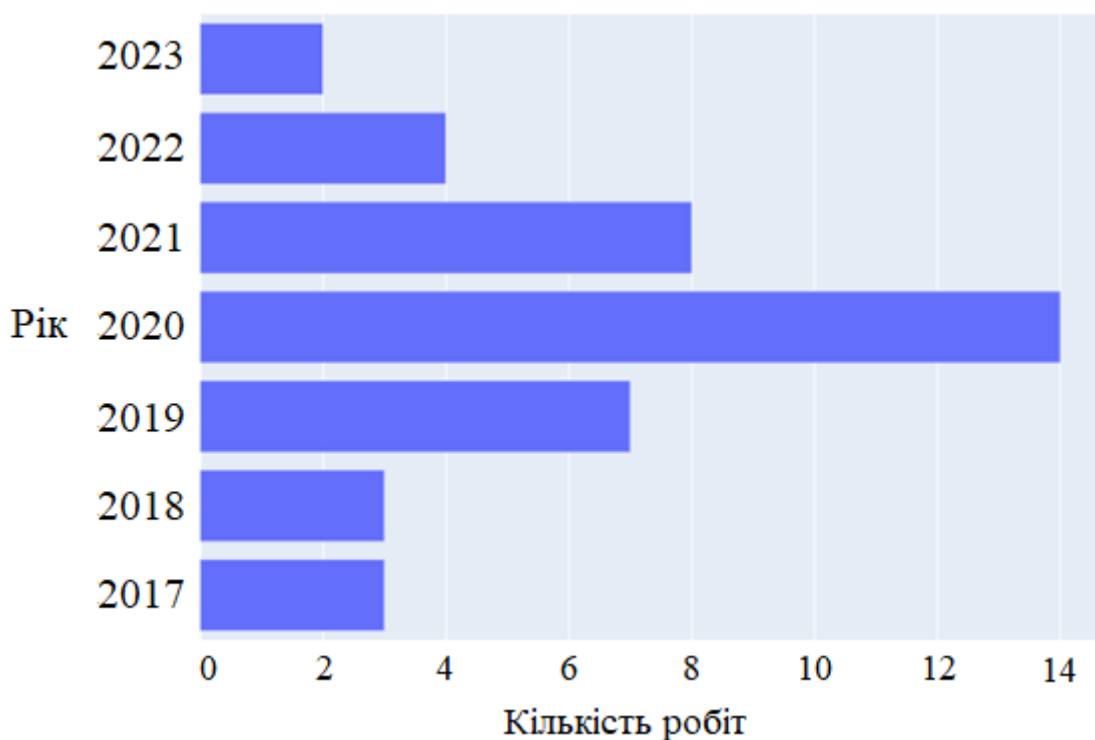


Рисунок 1.1 – Кількість знайдених робіт на тему розпізнавання захворювань за знімком КТ, розподілена по роках.

До 2020 року більшість досліджень, спрямованих на аналіз КТ-знімків легень фокусувались на пошуку та ідентифікації легеневих вузлів, включно з ідентифікацією злоякісних вузлів та пухлин.

Так, у [26] було використано поширену архітектуру згорткових нейронних мереж під назвою DetectNet, побудовану за принципом You Only Look Once (YOLO) [60], що часто застосовується для вирішення загальних задач з пошуку об'єктів на зображенні. YOLO розглядає задачу виявлення як єдину регресійну проблему, розділяючи зображення на сітку та передбачаючи обмежувальні рамки і ймовірності класів для кожної комірки сітки одночасно. Суттєвою перевагою YOLO у випадку легеневих вузлів є здатність не тільки класифікувати знімок як позитивний чи негативний, а й вказати розташування, на зображенні.

Також у дослідженнях до 2020 року розглядалась діагностика туберкульозу, емфіземи та пневмотораксу. Так, у [27] було використано комбінований метод, у якому КТ-знімок розбивався на окремі ділянки розміром 36*36 пікселів, які класифікувались окремо за допомогою згорткової нейронної мережі з метою пошуку

регіонів легень, імовірно уражених пневмотораксом. З результатів складались тривимірні теплокарти, які класифікувались за допомогою методу опорних векторів (SVM).

В 2020-му році вийшло найбільше статей на тему діагностики захворювань за допомогою аналізу знімків КТ легень – практично усі з яких з яких фокусувались на діагностиці COVID-19. Більшість з них класифікувала між двома класами – наявність або відсутність COVID-19.

Розглянемо деякі з них. Так, [35] є прикладом однієї з ранніх статей на тему автоматизованої діагностики COVID-19. У цій статті було представлено датасет COVID-CT, зібраний за допомогою зібрання 760 попередніх публікацій (препринтів) на medRxiv та bioRxiv. Зі статей було видобуто наявні в них зображення, виокремлено знімки КТ, та познімково визначено (за допомогою підпису рисунка та тексту статті) чи знімок являє собою зріз зображення легень хворого на COVID-19, а також зібрано інші метадані. До цих зображень було додано знімки здорових людей та пацієнтів з іншими хворобами легень, видобуті з відкритих датасетів, для утворення набору негативного класу. На основі отриманого датасету було натреновано DenseNet-169, та порівняно з аналогічною мережею, натренованою на меншому датасеті з чистих даних, отриманих з архівів госпіталів. Шляхом більшої різноманітності зображень, мережа на основі зібраного в дослідженні датасету перевершила аналогічну, досягнувши точності у 0.89, чого достатньо для клінічної корисності.

У [61] було порівняно архітектури VGG-16, ResNet-18, ResNet-50 та інші. Було визначено, що глибші мережі (ResNet-50 замість ResNet-18) загалом дають кращі результати класифікації. Однак, мережі з більшою кількістю вагів загалом (наприклад, VGG-16) вимагають використання передавального навчання для уникнення перевивчення датасету, не завжди дають кращі результати. Так, запропонована в дослідженні архітектура CRNet досягає F1 у 0.76, так само як і VGG-16, попри те, що має на три порядки меншу кількість вагів. Найкращі результати було досягнуто за допомогою ResNet-50, DenseNet-169 (F1-міра у 0.81).

У [45] було досягнуто F1 у 0.90 за допомогою нейронної мережі архітектури EfficientNet, натренованої на загальнодоступному наборі даних COVID-CT-Dataset.

Для цього набір даних було аугментовано горизонтальним та вертикальним віддзеркаленням зображень, та досліджено три методи керування коефіцієнтом швидкості навчання: постійний коефіцієнт, циклічне регулювання та метод плато (поступове зниження з наближенням до певної границі). Найкращий загальний результат (F1) було досягнуто за допомогою методу плато.

В [51] результати для ResNet-50 та VGG-16 було значно покращено. Ці архітектури було поєднано разом з мережею архітектури Xception у єдину складену мережу та утворено ансамбль моделей. Кожна з підмоделей отримує на вхід зображення та формує вектор результату класифікації. Обчислені вектори конкатенуються в єдиний та передаються метакласифікаторові, який приймає фінальне рішення. Утворений таким чином класифікатор, з певної точки зору, представляє складну модель що містить в собі базові. В процесі навчання базові моделі було навчено окремо одна від одної та метакласифікатору. Суб-модель VGG-16 отримала точність у 0.984, ResNet50 у 0.945, Xception у 0.962. Сукупний ансамбль отримав точність у 0.987.

Було визначено найбільш популярні архітектури нейронних мереж, застосовані до аналізу знімків КТ. Досягнуті ними точності, згідно з інформацією у літературних джерелах, вказано у таблиці 1.3.

Таблиця 1.3 – порівняння точностей для різних архітектур нейронних мереж

| Метод | DenseNet | ResNet | VGG | Xception | EfficientNet |
|--------------------|----------|-------------------------------------|-----------------------|------------------|------------------|
| Досягнуті точності | 88%, 83% | 93%, 97% 92.2%, 86%, 87%, 99% | 87%, 98%, 76%, 99% | 97%, 90%, 96% | 89%, 77%, 79% |

Розглянемо дані архітектури детальніше:

DenseNet [62] є архітектурою згорткових нейронних мереж, що використовує DenseBlock (щільні блоки), що (додатково до стандартних послідовних зв'язків) прямо поєднують між собою усі згорткові мережі з їх розмірностями, що дозволяє

поширювати вивчені ознаки між різними шарами мережі, та дозволяє зменшити розмір нейронної мережі.

ResNet [63] є архітектурою ЗШНМ, що застосовує перемички між шарами, що перестрибують один чи декілька шарів, що допомагає передачі інформації в глибші шари, що дозволяє уникати деяких проблем, що виникають при поглибленні (збільшенні кількості шарів) нейронної мережі.

VGG [64] є архітектурою ЗШНМ, що використовує послідовне звуження шарів, що дозволяє отримати глибші мережі при зменшенні чи збереженні кількості параметрів системи, та покращити розрізняючу здатність мережі.

Xception [65] є архітектурою ЗШНМ, що базується на використанні глибиннорозділеного згортання, яке, на відміну від типової згорткової операції, розділює обчислення на дві стадії – застосування глибинного згортання на кожний вхідний канал та точкове згортання для створення лінійного сполучення виходу глибинного згортання.

EfficientNet [66] є архітектурою ЗШНМ та методом їх масштабування, який рівномірно масштабує глибину, ширину та роздільну здатність мережі за допомогою складеного коефіцієнта. У такій архітектурі, на відміну від звичайних, ці параметри масштабуються не довільним шляхом, а набором конкретизованих коефіцієнтів [5].

З розглянутих робіт, 8 використовували виключно або в тому числі нейронні мережі архітектури ResNet, з середнім AUC в 95% та середньою точністю в 93%, що виділяє цю архітектуру як варту особливої уваги.

1.2.1 Проблеми, що пов'язані з аналізом результатів

Загалом, пряме порівняння між знайденими роботами є ускладненим двома факторами:

- проблеми з аналізом інформації визначення кількості знімків (тобто розміру датасет) на якому проводилось навчання/тестування;
- способи оцінки результатів класифікації.

Щодо проблеми визначення кількості знімків: роботи можуть бути спрямовані як на двовимірний, так і тривимірний аналіз – обробку окремих зрізів чи знімка

загалом. Відтак, розмір набору даних може бути як кількістю тривимірних знімків, так і кількістю окремих зрізів чи кількістю пацієнтів у дослідженні.

Також при представленні результатів досліджень автори використовують різні метрики оцінки досягнутих характеристик класифікації, що робить пряме порівняння між мережами ускладненим. З 39 робіт, 20 використовує точність (1.1), 11 використовує AUC ([59]), 4 використовує чутливість (sensitivity) (1.2) та специфічність (specificity) (1.3), 5 використовує F1-міру (1.4) та тільки одна використовує влучність (precision) (1.5) та повноту (recall) (1.6).

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1.1)$$

$$sensitivity = \frac{TP}{TP+FN} \quad (1.2)$$

$$specificity = \frac{TN}{TN+FP} \quad (1.3)$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (1.4)$$

$$precision = \frac{TP}{TP+FP} \quad (1.5)$$

$$recall = \frac{TP}{TP+FN} \quad (1.6)$$

– де TP – істинно позитивний, FP – хибно позитивний, TN – істинно негативний, FN – хибно негативний.

З врахуванням неповноти даних, що наведені у статтях, неможливо перерахувати результати у єдину шкалу виміру для порівняння.

Серед статей, що вийшли після початку пандемії COVID, більшість знайдених робіт були спрямовані на бінарну класифікацію з метою визначення наявності пневмонії, спричиненої COVID-19, та містять набір даних який включає лише знімки здорових легень та легень уражених пневмонією спричиненою COVID-19.

Результати аналізу літературних джерел виявили наступні особливості:

– відсутність у дискусії методів, що не є нейронними мережами;

- відсутність єдиної метрики результатів класифікації для порівняння;
- варіативність складу, розміру та методів опису наборів даних.

Таким чином, на основі проведеного аналізу, для покращення якості класифікації та значущості результатів можемо визначити наступні вимоги та параметри для систем класифікації захворювань:

- мінімальний розмір набору даних мусить становити щонайменше 200 пацієнтів (у тривимірних зображеннях DICOM, або еквівалентній кількості послідовних двовимірних зрізів не менше 30 на пацієнта);
- для мультикласової класифікації всі класи повинні рівномірно представлені в наборі даних;
- стандартні метрики не повинні ігноруватися, аби результати були порівнювані з існуючими дослідженнями.

Також слід зазначити, що особливу увагу варто приділяти відтворенню та вдосконаленню таких робіт, як Hoop Ko et al. [34], Xuehai He et al. [67], Qianqian Ni et al. [38], Min Fu et al. [40], Xin He et al. [42], Matthias Fontanellaz et al. [46], Mehdi Yousefzadeh et al. [47]. Ці роботи використовують дані, які включають інші захворювання легень, окрім цільового захворювання, — найчастіше третім класом у дослідженні є позагоспітальна пневмонія. Мережі, навчені на таких наборах даних, є більш стабільними та більш придатними для реального застосування і адаптації до інших захворювань, включно з новими через їх різноманітність.

1.3 Використання потоків робіт у аналізі зображень для діагностики

Процес розробки програмного забезпечення на основі нейронних мереж може бути розділений на наступні кроки:

1. проектування та тренування нейронної мережі;
2. інтеграція нейронної мережі у ПЗ.

Якщо перший крок є широко обговорюваним щодо виконання діагностики за допомогою аналізу зображень нейронними мережами, другий крок у літературі

практично не зустрічається. Відтак, розглянемо питання вбудування НМ у програмне забезпечення.

Спосіб інтеграції повинен адаптуватися до раптових змін у навантаженні на систему у разі спалаху певного захворювання. Одним з виходів з даної ситуації є підтримка розподілених обчислень та масштабованості програмного забезпечення.

Додатково, важливим аспектом засобу для діагностики є швидкість адаптації системи до появи нових захворювань. Відтак, є потреба у зменшенні зусиль, необхідних для додавання чи модифікацію методу діагностики, який виконується у засобі.

Найпростішим масштабованим способом розгортання нейронних мереж для використання у програмних засобах є створення відповідних мікросервісів, до яких звертаються інші компоненти системи, шляхом HTTP, gRPC чи інших запитів (рис. 1.2).

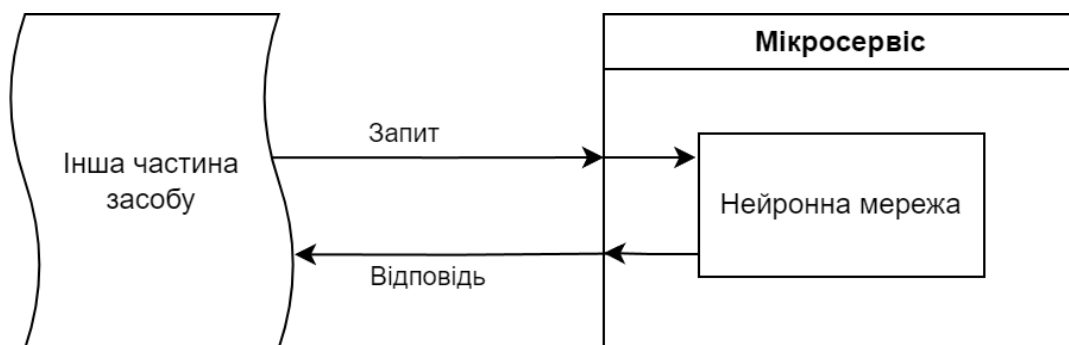


Рисунок 1.2 – Типовий спосіб використання нейронних мереж у хмарних ПЗ

Головною перевагою такого способу є його відносна простота, яка виражається у швидкості первинної розробки системи. Однак, такий шлях доволі жорстко фіксує складові частини, що виконують аналіз даних. Це проявляється у тому, що класифікатор:

- жорстко прив’язується до механізмів, що інтегрують його у решту системи;
- збирається, постачається та розгортається виключно разом з цими механізмами.

Відтак, для модифікації, заміни чи доповнення класифікатора необхідним є втручання у код мікросервісу та його перерозгортання, що ускладнює та сповільнює

модифікацію засобу аналізу зображень, а також обмежує коло людей що може здійснити таку модифікацію.

Альтернативним підходом є винесення нейронної мережі в окрему сутність, що комунікуватиме з рештою системи на примітивному рівні. Наприклад, це може бути виконано у якості контейнеризованого скрипту, що запускається з певними вхідними даними у вигляді файлу(-ів), здійснює їх обробку, та зберігає результат у вихідний(-і) файл(-и) (рис. 1.3).

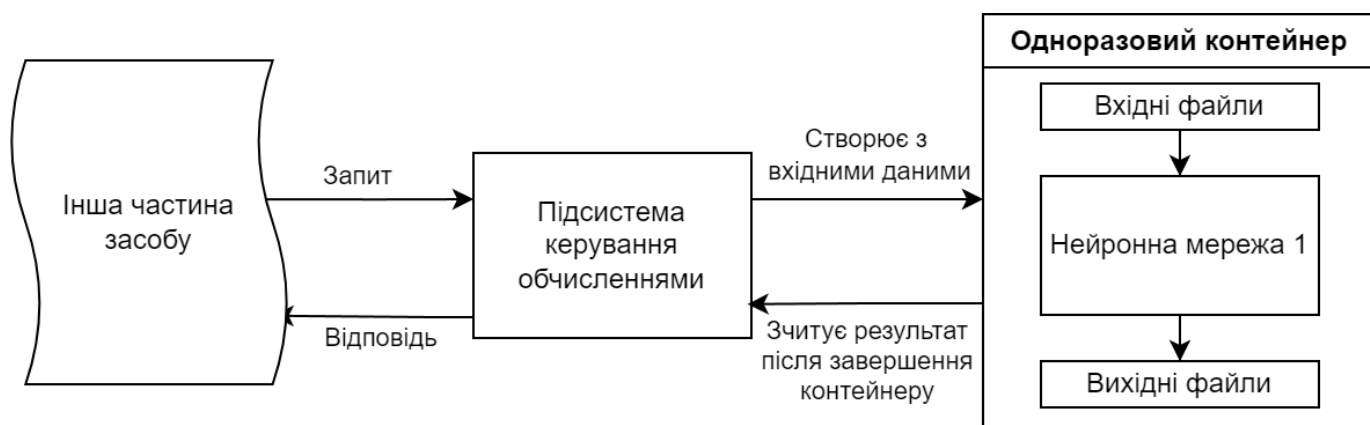


Рисунок 1.3 – Ізоляція нейронних мереж в одноразових контейнерах

Такий підхід дозволяє модифікацію (дотренування) наявних та додавання нових методів класифікації та нового аналітичного функціоналу (підтримка інших хвороб чи принципово інших вхідних даних) без втручання у механізми, що забезпечують функціонування системи та комунікацію між її компонентами.

Таким чином досягається спрощення та пришвидшення адаптації засобу загалом, у тому числі робить можливим модифікацію засобу силами працівників лабораторії, що досліджує джерело інфекції з мінімальним або відсутнім залученням інженерів з розробки програмного забезпечення.

Одним з способів реалізації такого підходу є використання систем керування потоками робіт (СКПР).

Потоки робіт (ПР) є впорядкованими структурованими представленнями багатокрокових обчислювальних задач. Типовий ПР може бути представлений за допомогою спрямованого ациклічного графа. В графовому представленні, вершини є

певними операціями над даними, що мають один чи декілька входів. Ребра ж відмічають «перетікання» виходів одних операцій у входи інших. Такий граф, що складається з вхідних даних, та їх послідовних перетворень за допомогою окремих вершин (також – кроків або інструментів), часто називається пайплайном (англ. pipeline).

Інструменти обробки даних, які є вершинами графу потоку робіт, залежно від системи, можуть бути реалізовані як модулі в певній мові програмування, наприклад R чи Python, так і бути окремими програмами з інтерфейсом командного рядка.

З поширенням Docker [68] та інших систем контейнеризації програмного забезпечення, все більше СКПР спирається саме на загорнуті в контейнери інструменти для виконання обчислень. Завдяки контейнеризації обчислювальних інструментів досягається ізоляція контекстів виконання. Окрім іншого, це має особливе значення для залежностей інструментів, оскільки в багатьох сферах аналіз даних вимагає великої кількості різноманітних компонент. Ці компоненти можуть бути реалізовані за допомогою різних мов програмування та бібліотек, та можуть розроблятися як всередині команди що проводить дослідження або розробляє інструмент, так і третіми сторонами.

Так, використання виключно інструментів з інтерфейсом командного рядка:

- вимагає стандартизації списку залежностей всього потоку робіт;
- вимагає встановлення залежностей у оточення, у якому виконуються обчислення;
- ускладнює використання інструментів з несумісними залежностями.

Натомість використання контейнеризованих інструментів дозволяє:

- ізолювати інструменти та їх залежності одне від одного;
- постачати залежності інструментів разом з інструментами;
- гарантувати використання залежностей точних версій, з якими інструмент протестовано.

Загалом, упакування інструментів у спеціалізовані контейнери, що містять все для виконання задачі інструменту, спрощують розробку, розгортання та підтримку систем, що використовують ці інструменти.

СКПР дозволяють визначити використовувані інструменти, їх входи та виходи, та поєднувати їх у потоки робіт. Ці ПР можуть бути запущені як на одиничних машинах – наприклад, персональному комп'ютері інженера чи науковця який займається розробкою потоку робіт, або обчислювальному сервері, так і на кластерних чи грид-системах, в тому числі на хмарних обчислювальних потужностях.

Таким чином керування ресурсами, розміщення та передача даних, планування роботи інструментів тощо абстрагується у простіший та доступніший інтерфейс для проведення розподілених обчислень.

В наукових, а особливо біоінформатичних, дослідженнях потоки робіт та системи керування ними широко застосовуються для проведення обчислювальних експериментів над здобутими даними, доведення гіпотез тощо. Використання саме таких засобів дозволяє вченим уникнути вирішення відносно низькорівневих задач планування ресурсів та керування ними, паралелізації обчислень, керування розміщенням та передачею даних тощо, делегуючи ці задачі системі керування потоками робіт. Також, СКПР надають можливість інтерактивно спостерігати за виконанням задач у потоці, відстежувати історію перетворень елементів даних, та перевикористовувати як окремі інструменти, так і цілі потоки робіт.

Таким чином, використання СКПР для виконання діагностики захворювань за допомогою класифікації зображень є доцільним. Варто звернути увагу, що їх використання не є обов'язковим для досягнення необхідної мети, але дозволяє спростити та пришвидшити модифікацію системи.

1.4 Засоби для керування потоками робіт

Згідно зі списком [69], що підтримується спільнотою Common Workflow Language, наразі налічується щонайменше 348 різних систем керування потоками робіт. Ці системи мають різні призначення та обмеження, спричинені специфікою передбачених сфери та умов застосування, однак деякі з цих систем набули широкого поширення.

При виборі СКПР, ключовими питаннями для нас є:

- Які способи запуску інструментів підтримуються системою (модулі специфічної мови програмування, bash-команди, docker-контейнери);
- Чи є підтримка розподілених/кластерних обчислень;
- Яким чином визначаються потоки робіт, наскільки простим є програмування потоку робіт;
- Чи є можливість реалізації умовних переходів (запуску тих чи інших інструментів залежно від певних умов).

1.4.1 Apache Airflow

Apache Airflow є системою керування потоками робіт з відкритим початковим кодом. Оператори (інструменти) Apache Airflow можуть бути як і bash-командами чи python-програмами, так і запускатись у контейнеризованому вигляді як Kubernetes-поди. Також, Apache Airflow дозволяє написання власних видів операторів для інтеграції довільних мов програмування, способів виконання коду чи розподілених систем.

Потік робіт в Airflow є спрямованим ациклічним графом. Оператори, задачі та їх залежності визначаються розробником потоку робіт у вигляді Python-коду. Так, вид операторів (python, bash, Kubernetes) є класом. Для запуску оператору розробник мусить інстанціювати його, створивши таким чином задачу. При створенні задачі розробник задає умови виконання задачі та її зміст. Для створення спрямованого ациклічного графа визначаються залежності та послідовності передачі даних між задачами.

Створені спрямовані ациклічні графи реєструються в системі та можуть бути запуснені як у відповідь на певну подію – від запуску користувачем вручну у графічному інтерфейсі до появи нового файлу у сховищі даних (такому як Apache HIVE), так і регулярно – щогодинно, щоденно, щотижнево тощо. Apache Airflow може бути розгорнутий як на локальній машині розробника, так і у контейнеризованих, кластерних та грид-системах.

Apache airflow широко застосовується як для виконання внутрішніх аналітичних обчислень у комерційному середовищі, так і для проведення наукових та

інших обчислень. Гнучкість у розгортанні та інтеграції з джерелами та сховищами даних значно полегшує та пришвидшує розробку масштабованих та дата-інтенсивних систем.

1.4.2 Nextflow

Nextflow [70] є системою керування потоками робіт з відкритим початковим кодом. Процеси (інструменти), канали та потоки робіт Nextflow визначаються за допомогою предметно-орієнтованої мови програмування, яка є розширенням мови програмування Groovy для віртуальної машини Java. Для процесу визначаються вхідні та вихідні дані, скрипт-програма на bash, groovy чи довільній скриптовій мові програмування (викликається за допомогою інтерпретаторної директиви -- шебангу), необхідні умови виконання, середовище (наприклад, Docker-зображення), тощо.

Канали у Nextflow є потоками даних, за допомогою яких потік робіт отримує вхідні дані, а процеси всередині потоку робіт можуть ними обмінюватись. Канали та дані всередині них можуть бути оброблені різноманітними операторами – від фільтрації елементів даних та читання CSV та JSON, до розділення потоку до різних процесів, та поєднання різних потоків в один для подальшої обробки процесом.

Потік робіт у Nextflow є композицією процесів та каналів. У спеціальній конструкції скриптові мови Nextflow розробник описує необхідні потоки даних та викликає потрібні процеси, задаючи їм вхідні дані та передаючи їх вихідні дані до входів інших процесів або до результату виконання потоку робіт. Під час запуску потоку робіт, Nextflow виконуватиме процеси як тільки будуть задовільнені їх умови запуску, за можливості виконуючи їх паралельно.

Потоки робіт Nextflow можуть бути виконані у різноманітних середовищах – як на локальній машині, з використанням локального docker для запуску контейнеризованих процесів, так і в хмарних, кластерних та ґрід-середовищах, у тому числі з використанням Kubernetes та менеджерів ґрід-систем.

Nextflow широко застосовується у наукових обчисленнях у сфері біоінформатики для проведення геномних, протеомних обчислень, *in silico* дослідженні медичних препаратів, тощо.

1.4.3 Системи на основі Common Workflow Language

Common Workflow Language (CWL) [31] є відкритим стандартом для опису потоків робіт. CWL був розроблений для уможливлення створення потоків робіт для аналізу даних, з акцентом на портативність (запуск у різноманітних системах керування потоками робіт) та можливість повторного використання ПР та їх елементів, та широко застосовується в біоінформатиці, астрономії, хімії, аналізі медичних зображень.

У CWL інструменти та потоки робіт визначаються у вигляді YAML чи JSON-файлів, які містять опис необхідних інструментів, та набір кроків, кожен з яких є викликом певного інструменту з певними вхідними даними та передачею виходів цього інструменту як вхідних даних для інших кроків. Таким чином, кроки – виконання інструментів з певними параметрами та вхідними даними – поєднуються у повноцінний спрямований ациклічний граф. CWL-інструменти можуть бути Javascript-виразами або програмами командного рядка, що запускаються на хост-машині чи в середовищі docker-контейнеру. Також, потік робіт може містити виклик іншого потоку робіт як інструменту.

На відміну від Airflow та Nextflow, у Common Workflow Language потоки робіт є відносно статичними, оскільки єдина опція надана CWL для вкладення логіки для прийняття рішень потоком робіт поза його інструментами полягає у існуванні кроків з умовою запуску.

Оскільки Common Workflow Language є лише стандартом описання потоків робіт, для їх запуску необхідно використати певний СКПР чи платформу. Існує багато способів запустити потік робіт, реалізований на CWL, серед них варто виділити наступні:

- cwltool – еталонна реалізація стандарту CWL. Реалізована мовою програмування Python, та дає можливість виконувати потоки робіт на одиничній машині за допомогою локального Docker. Також, cwltool надає Python-інтерфейс, який дозволяє обробляти CWL-документи, запускати потоки робіт локально, а також – запускати потоки робіт CWL на довільних

обчислювальних потужностях за умови ручної інтеграції цих потужностей за допомогою передачі cwltool ряду функцій, які відповідають певному інтерфейсу;

- Toil [71] – масштабована реалізація CWL та WDL (Workflow Description Language), що може бути запущена як на локальній машині, так і виконувати обчислення на AWS та Google Cloud прямою роботою з їх API чи допомогою Kubernetes. Однак, підтримка Kubernetes наразі є обмеженою, оскільки вимагає AWS Job Store для зберігання даних. Також, Toil підтримує проведення обчислень у ґрід-системах;
- REANA [72] є масштабованою реалізацією CWL, Serial, Snakemake та Yadage, розробленою у CERN. Підтримує розгортання та обчислення за допомогою Kubernetes та ґрід-систем Slurm та HTCondor.
- Arvados [73] є реалізацією CWL, першочергово орієнтованою на обробку великих даних. Arvados має вбудовану систему зберігання масивних об’ємів даних, та може бути розгорнутий як на основі AWS та Microsoft Azure, так і на власних потужностях за допомогою Slurm та IBM Spectrum LSF.

1.4.4 Порівняння систем керування потоками робіт

На початку підрозділу 1.4 було визначено ряд критеріїв для оцінюваних систем. У таблиці N наведено порівняння між розглянутими системами за цими критеріями.

Таблиця 1.4 – Порівняння систем керування потоками робіт

| Система | Підтримувані інструменти | Підтримка кластерних обчислень | Визначення потоків робіт | Умовні переходи |
|----------------|--|--------------------------------|---|-----------------|
| Apache Airflow | Bash-команди, Python-програми Docker-контейнери | Kubernetes Грід-системи | У мові програмування Python | Так |
| Nextflow | Bash-команди Groovy-програми Скриптові програми Docker-контейнери | Kubernetes Грід-системи | Предметно-орієнтована мова програмування на основі Groovy | Ні |
| Cwltool | Bash-команди JS-скрипти Docker-контейнери | Відсутня | CWL (JSON/YAML) | Ні |

| | | | | |
|---------|---|----------------------------|-----------------|----|
| Toil | Bash-команди JS-скрипти Docker-контейнери | Kubernetes Грід-системи | CWL (JSON/YAML) | Ні |
| REANA | Bash-команди JS-скрипти Docker-контейнери | Kubernetes Грід-системи | CWL (JSON/YAML) | Ні |
| Arvados | Bash-команди JS-скрипти Docker-контейнери | Kubernetes Грід-системи | CWL (JSON/YAML) | Ні |

Аналіз порівняльної таблиці демонструє, що існуючі СКІР забезпечують широкий спектр можливостей. Усі розглянуті системи підтримують контейнеризацію, що забезпечує відтворюваність обчислень та спрощує розгортання. Більшість систем надають засоби для роботи у кластерних та грід-середовищах, включно з Kubernetes та спеціалізованими системами керування обчислювальними ресурсами.

Однак, спільним обмеженням усіх розглянутих систем є статична природа визначення потоків робіт. Незалежно від обраного методу, граф потоку робіт має бути явно визначений розробником заздалегідь. При внесенні змін до окремого інструменту обробки даних у потоці необхідним є:

- визначити усі ПР, що використовують цей оператор;
- оновити визначення кожного такого потоку;
- перерозгорнути оновлені ПР.

Такий підхід створює накладні витрати часу при впровадженні змін та вимагає додаткових зусиль для підтримки узгодженості між різними потоками. Відтак, перспективною є розробка підходу та системи, які дозволили б динамічно створювати ПР на основі наявних вхідних даних та наявних інструментів їх обробки. У такій системі, оновлення інструменту автоматично відображалось б у всіх потоках, що його використовують, без необхідності їх явного оновлення чи перевизначення.

1.5 Висновки до розділу

Сучасний стан аналізу зображень КТ для діагностики демонструє наявність засобів та методів класифікації найбільш розповсюджених захворювань. Основні

зусилля наукової спільноти спрямовані на покращення результату визначення хвороби за допомогою інструментарію нейронних мереж. Існуючі результати розробки класифікаторів, здебільшого бінарних, показують виключно можливість розпізнавання певних характерних ознак хвороби, як звичайних елементів зображення.

Однак, у випадку епідемій або пандемій, критичним є якнайшвидше охоплення діагностикою великої кількості пацієнтів та час реакції систем охорони здоров'я. Наявні результати аналізу літературних джерел показують, що в дослідженнях даним аспектам майже не приділяється уваги.

Швидкість реакції системи охорони здоров'я залежить від декількох компонентів (персонал, матеріально-технічне забезпечення, логістика тощо), в тому числі і швидкість адаптації існуючих засобів діагностики та ПЗ до нових викликів та задач.

Існуючі інструменти хмарних обчислень та можливість дистанційного оновлення ПЗ дають можливість розробки та впровадження нових підходів щодо архітектури систем діагностики та побудови класифікаторів. Окрім необхідності передбачити донавчання існуючих або використання додаткових класифікаторів, критичною може стати необхідність суттєвих змін у системі, що може потребувати втручання спеціалізованих фахівців (наприклад, програмістів), що значно збільшує час готовності до використання.

Існуючі системи керування потоками робіт вимагають статичного визначення графу потоку робіт розробником системи. Існування даних статичних визначень уповільнює внесення змін. Відтак, перспективним є розробка архітектури, методів та програмних засобів, що динамічно створюють та обчислюють потоки робіт виходячи з наданих даних та інструментів для їх обробки та перетворення, а також дозволяють розширювати існуючі засоби класифікації захворювань без втручання або мінімальним втручанням людини (фахівця).

2 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ ДЛЯ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ

Існує багато типів медичних візуальних даних, від фотографій уражень шкіри до радіограм та МРТ-знімків. Проте обсяг вільно доступних даних значно відрізняється залежно від типу зображень. Для розробки ефективних діагностичних інструментів на основі методів машинного навчання важливо забезпечити наявність достатньої кількості якісних даних у тренувальному наборі. Це є ключовим фактором для досягнення високої точності моделей.

Збір медичної інформації для дослідницьких цілей у більшості країн є багатокроковим процесом. Зокрема, зібрані дані повинні пройти ряд процедур для забезпечення захисту прав пацієнтів як у самому наборі даних, так і у дослідженнях, що на ньому базуються. Під час пандемії COVID-19 широкому колу науковців було надано доступ до даних з метою поєднання зусиль спеціалістів різних галузей для боротьби з хворобою. Дослідницькими колективами було сформовано низку датасетів, в тому числі і такі, що склались зі знімків КТ. Загальнодоступні дані дають змогу зосередитись на науковій складовій задач, що вирішуються, та зекономити час на вирішення юридичних та організаційних питань.

Відтак, зважаючи на проблеми зі збором вирішено використати наявні відкриті набори даних, що фокусуються на зображеннях, отриманих за допомогою комп'ютерної томографії.

2.1 Формування набору даних

За основний було взято відкритий набір даних COVID-CTset (набір А) [74]. Набір А був зібраний в медичному центрі Негін (Сарі, Іран) за допомогою сканера SOMATOM та програмним забезпеченням syngo CT VC30-easyIQ. Зображення постачаються у вигляді окремих почергових зрізів знімків КТ, у форматі TIFF, чорно-білі, з 16-бітною глибиною кольору. Датасет складається з зображень, що належать 95 пацієнтам з діагностованим COVID-19 та 282 здоровим людям.

Формат TIFF (Tagged Image File Format) є одним зі стандартних для збереження зображень, розроблений для задач поліграфії. Оскільки формат є гнучким, підтримує

різні колірні моделі та розрядності, а також дозволяє стиснення без втрат, його також використовують для зберігання та аналізу зображень для машинного навчання та інших цілей. Однак, А містить тільки два класи, внаслідок чого було додано набір даних COVID-CT-MD [75] (набір Б).

Набір Б було зібрано у медичному центрі Бабак (Тегеран, Іран) за допомогою сканера SOMATOM. Зображення представлені у вигляді почергових зрізів знімків КТ, у старому варіанті формату DICOM з 16-бітною глибиною кольору. Він містить зображення, що належать 169 людям з діагностованим COVID-19, 76 здоровим людям та 60 людям з діагностованою позагоспітальною пневмонією.

DICOM (Digital Imaging and Communications in Medicine) є стандартом для зберігання та передачі медичних зображень та пов'язаної з ними інформації, який включає формат для кодування зображень. DICOM широко використовується медичним обладнанням та програмним забезпеченням.

Для уніфікації наборів даних зображення з набору Б було перетворено у 16-бітний формат TIFF за допомогою модифікованого коду бібліотеки dcm2hdr і додано до набору А. Використання 16-бітних чорно-білих TIFF-зображень поєднує переваги формату DICOM і стандартних форматів зображень, дозволяючи зберегти повну глибину кольору та всі деталі оригінальних КТ-знімків, водночас забезпечуючи зручність обробки, властиву стандартним графічним форматам.

Відтак, було сформовано набір даних, що складається зі 264 знімків людей з COVID-19, 358 з здоровими легенями та 60 з позагоспітальною пневмонією.

Знімки КТ легень найчастіше виконуються починаючи трошки вище верхівки легень (біля яремної ямки) та охоплюють близько 25 сантиметрів вниз до місця з'єднання між діафрагмою та задньою частиною грудної клітини [76]. Окрім легень, на знімках також видно зрізи інших частин тіла, розташованих поруч із легенями. Таким чином, не усі зрізи знімка КТ є корисними для виконання діагностики. На рис. 2.1 представлено відмінність між зображеннями з різних частин тривимірного КТ-знімка: початковий зріз знімка, який не містить легені (1) та зріз що містить основну частину легень (2).

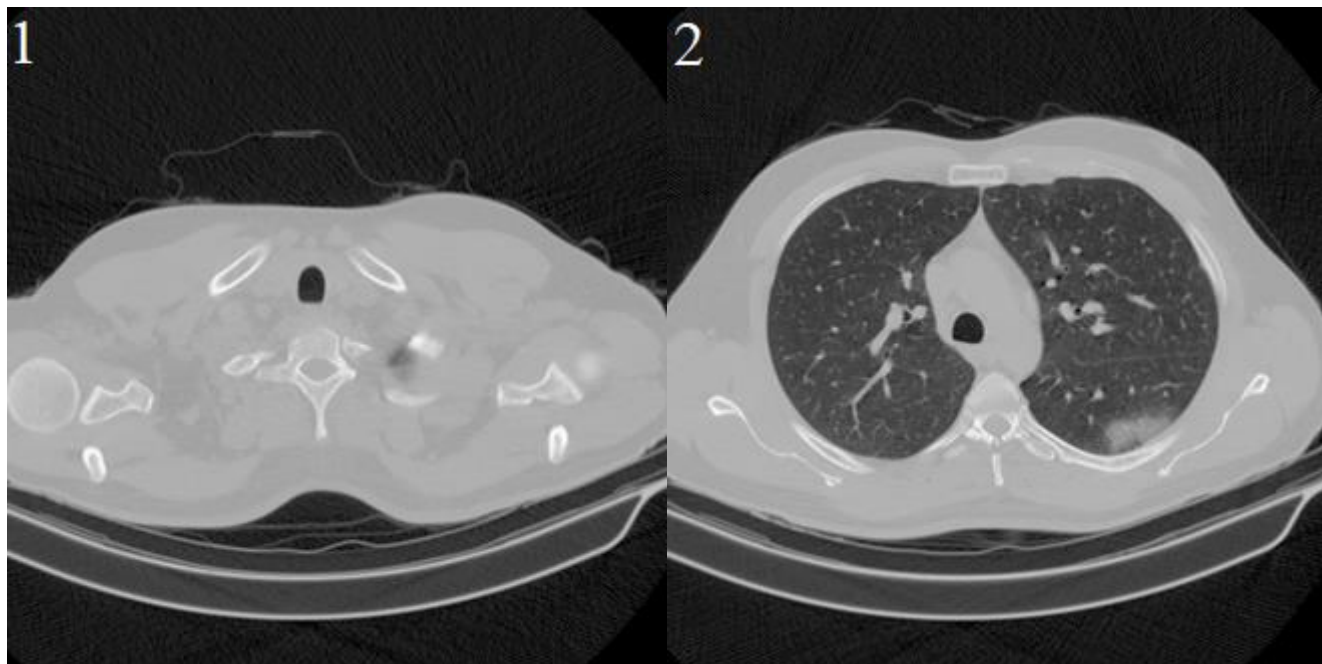


Рисунок 2.1 – Порівняння знімків КТ грудної клітини на різних зрізах просканованого простору

Датасет, що використовується у даній роботі складається з знімків, відфільтрованих за наявністю легенів по аналогії з алгоритмом, наведеним у [74] та використаним у наборі даних А. Фільтрацію зображень виконано шляхом перевірки змісту регіону в центрі зображення – якщо регіон переважно темний – зображення містить легені, в іншому випадку – відображає зону тіла, що передує або слідує легеням.

2.2 Мультикласова класифікація знімків КТ

Під час спалаху нового або малодослідженого захворювання діагностичні системи стикаються з проблемою, оскільки неможливо заздалегідь підготувати їх до розпізнавання нових патологій, прояви яких можуть бути невідомі. Це вимагає термінового розширення системи класифікації зображень для включення нових класів захворювань. Такий процес зазвичай вимагає зміни топології нейронної мережі та її повторного навчання на нових даних, що потребує значного втручання розробника і може зайняти чимало часу. В умовах кризових ситуацій, таких як раптовий спалах захворювання, критично важливо мінімізувати цей час, оскільки

швидка адаптація діагностичних засобів є одним із ключових факторів, необхідних для ефективного контролю спалаху.

Розглянемо модифікацію згорткової нейронної мережі бінарної класифікації для підтримки трьох класів. За базову реалізацію будемо вважати ЗНМ, розроблену для роботи з двовимірними зрізами знімків КТ в [74], де розглядалось розрізнення тільки між двома класами – «здоровий» або «COVID-19», що суттєво обмежує практичне застосування.

В основі базової ЗНМ лежить архітектура ResNet50V2. Оскільки прояви як типової пневмонії, так і пневмонії, спричиненої COVID-19, мають суттєві відмінності в характеристиках і масштабах, у [74] було запропоновано використати Feature Pyramid Network (FPN) [77].

FPN дозволяє покращити здатність мереж виявляти, вивчати та розпізнавати ознаки, текстури та об'єкти, пов'язані з ураженням легень. Такий підхід дозволяє досягти кращої точності класифікації за допомогою введення ієрархії ознак – починаючи, в цьому випадку, з третього шару (рис. 2.2) мережі ResNet результат роботи кожного згорткового шару мережі передається до додаткових згорткових шарів що оброблюють інформацію з основних шарів у зворотному порядку. Таким чином кінцеві щільні шари, що приймають рішення мають ширший доступ до інформації з первинної згорткової мережі, що дає можливість розпізнавати об'єкти різного масштабу та абстрактності. Дана властивість є особливо критично для медичних зображень, де діагностичні ознаки можуть суттєво відрізнятися за розміром.

Можливість розширення даної ЗНМ новим класом перевіримо на прикладі модифікації її топології для підтримки розрізнення між трьома класами: COVID-19, позагоспітальною пневмонією та здоровими легенями (рис 2.2).

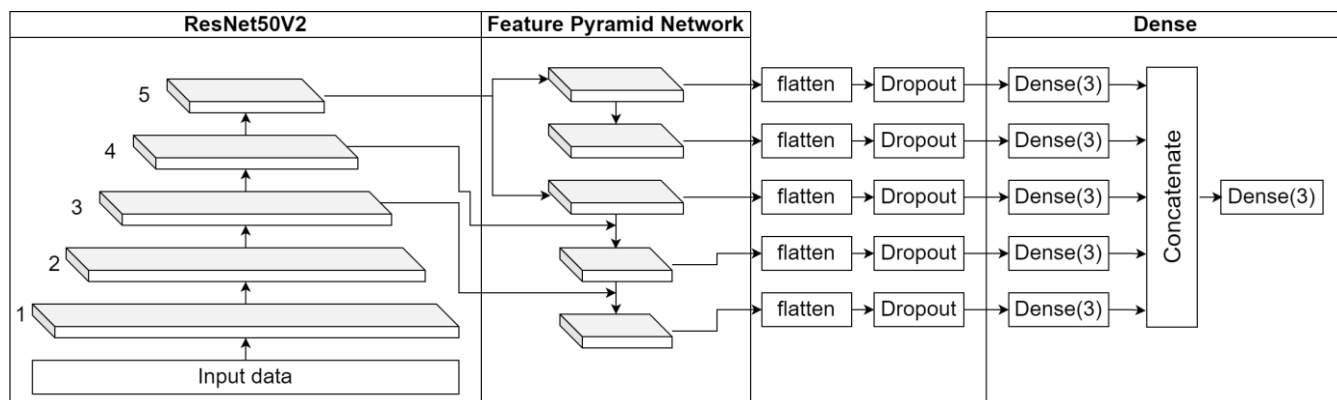


Рисунок 2.2 – Нейронна мережа для мультикласової класифікації

Для тренування нейронної мережі використано стартові ваги аналогічної нейронної мережі, натренованої на датасеті ImageNet [78]. Дані, що є відібраними з датасетів зображеннями, які містять відкриті легені, було розділено на наступні групи:

- 20701 тренувальних;
- 6900 валідаційних;
- 6900 оціночних.

Тренування було проведено протягом 20 епох. В кінці кожної епохи точність нейронної мережі було оцінено за допомогою валідаційного набору даних. Ваги НМ, точність яких була вищою за попередні, зберігались.

За результатом тренування було виявлено, що найкращу точність (94.96%) на валідаційному наборі даних було досягнуто на кінець 13 епохи навчання. Отриманий результат було перевірено за допомогою тестувального набору даних, та отримано точність у 95.086%. Близькість значень точності на валідаційному та тестувальному наборах (різниця у ~0.13%) вказує на здатність моделі до узагальнення ознак.

Для представлення результатів, зазвичай, також використовується матриця невідповідності, яка представляє порівняння фактичних результатів класифікації з істинними класами даних. За допомогою неї можна оцінити як часто модель помиляється, та побачити, дані яких класів модель класифікує хибно частіше ніж інші. Матрицю невідповідності для даного експерименту наведено на рис. 2.3.

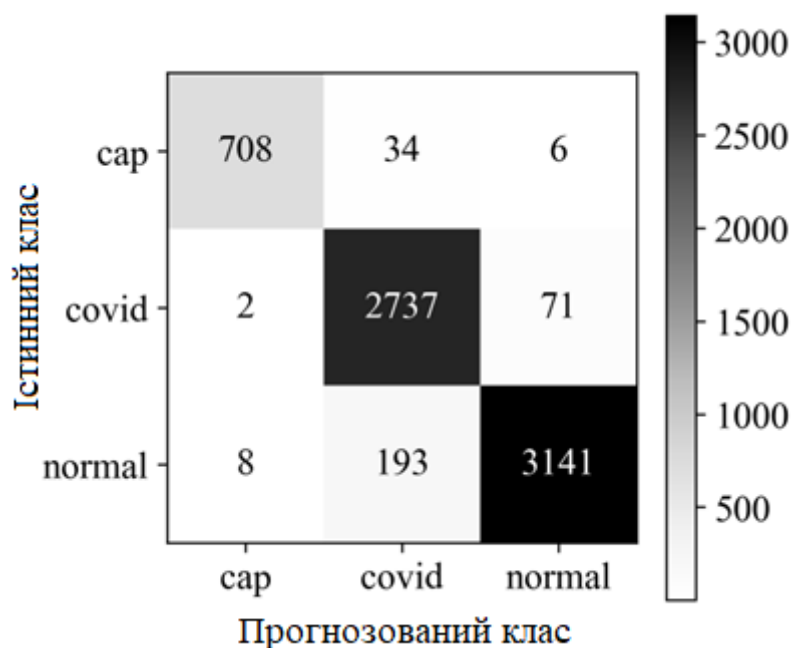


Рисунок 2.3 – Матриця невідповідності мультикласової НМ

У цьому випадку нейронна мережа демонструє значно більше хибно позитивних результатів порівняно з хибно негативними. Така ситуація є прийнятною для медичного застосування, оскільки пріоритетним є виявлення хворих пацієнтів, навіть якщо це призведе до помилкових підозр щодо здорових.

Наведений експеримент показує, що типові моделі бінарної класифікації можуть бути розширені без суттєвої (~3%) втрати якості класифікації. Недоліком даного підходу є необхідність зміни топології НМ і, відповідно, втручання в вхідний код, а також її перенавчання. Тому для мінімізації часу, необхідного на інтеграцію підтримки нових захворювань у діагностичні засоби доцільним є пошук методів класифікації, які дозволятимуть ефективно та швидко додавати нові класи через оновлення набору даних і донавчання мережі.

2.3 Використання вкладених представлень для класифікації зображень

Для роботи з текстами та зображеннями часто використовується такий підхід, як кодування інформації у скороченому векторному вигляді шляхом побудови вкладених представлень (ВП) (англ. «embedding»). Вкладені представлення є відображеннями складних даних (наприклад тексту чи зображення) у певному

векторному просторі [79] що зберігають ключову семантичну інформацію даних. У випадку зображень, такою семантичною інформацією є:

- колір;
- текстури;
- форми тощо.

Таким чином, ВП містить видобуту з зображення інформацію, та розміщується у певному векторному просторі, де представлення схожих зображень є ближчими за представлення відмінних зображень. Варто також зазначити, що розмірність векторного представлення як правило є суттєво меншою за розмірність репрезентованих даних, відтак, вкладені представлення у тому числі є методом зниження розмірності.

В цьому випадку, цікавим є використати даний підхід до знімків КТ, що дозволить, після початкового навчання нейронної мережі, за появи необхідності у підтримці нових класів донавчити НМ без зміни топології. У такому випадку, для виконання класифікації на основі ВП знімка використовуватимуться інші алгоритми машинного навчання, які можуть бути перетреновані для підтримки більшої кількості класів без втручання у код та за відносно короткий час.

Для того, щоб перевірити можливість використання ВП для зображень КТ, пропонується використати наявну класифікаційну ЗНМ для побудови вкладених представлень.

2.3.1 Побудова вкладених представлень зображень та виконання класифікації за ними

Запропонований процес побудови класифікаційної системи на основі вкладених представлень складається з наступних кроків:

1. тренування класифікаційної згорткової нейронної мережі;
2. використання ваг згорткової частини НМ (п.1) для тренування генератора вкладених представлень;
3. навчання алгоритмів для класифікації на основі згенерованих вкладених представлень;

Крок 1. Реалізується згорткова мережа для класифікації зображень, за прикладом мережі, модифікацію якої наведено у розділі 2.2 для мультикласової класифікації зображень КТ легень (рис. 2.4, А).

Мережа складається з трьох основних секцій. Вхідні дані обробляються стандартною моделлю ResNet50V2. Ідентично до мережі у 2.2, інформація як з вихідного, так і з внутрішніх шарів ResNet повторно обробляються за допомогою Feature Pyramid Network [77] з метою полегшення виявлення ознак різного розміру й абстрактності, від деталей текстури до загальних форм. Ознаки, видобуті за допомогою FPN, використовуються повнозв'язними шарами для прийняття остаточного рішення щодо класифікації.

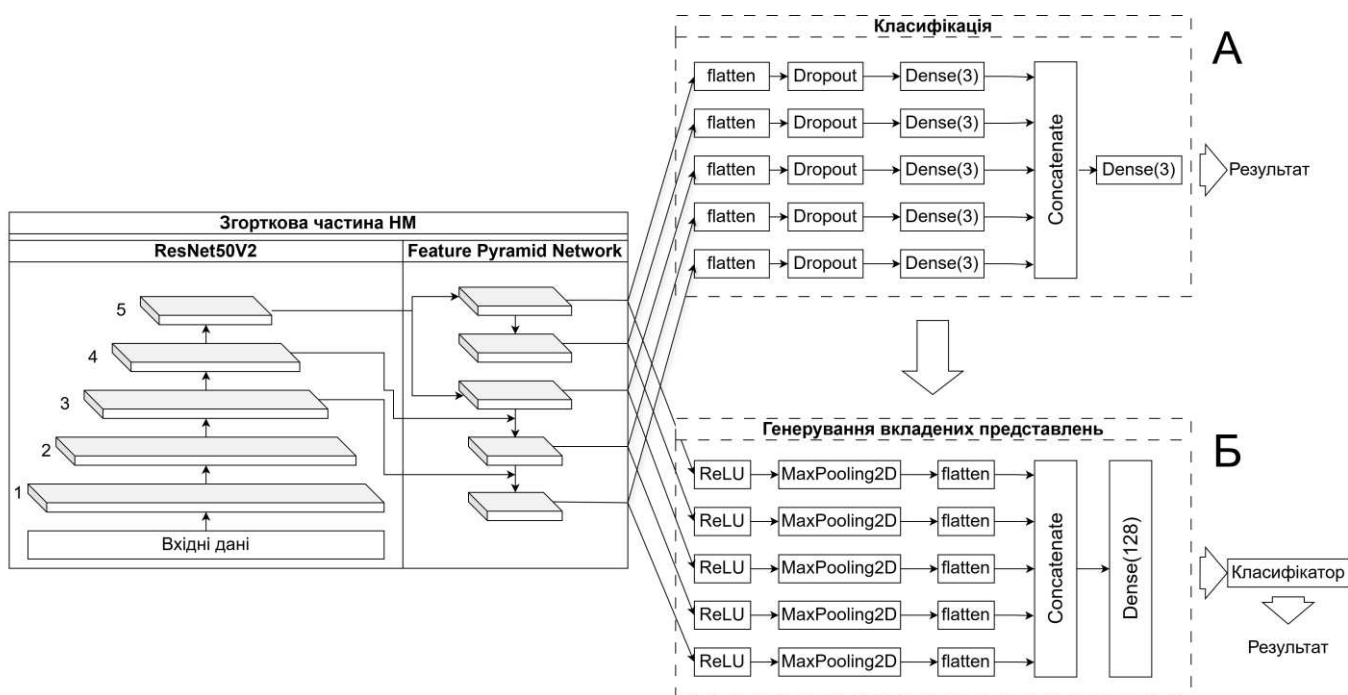


Рисунок 2.4 – Демонстрація відмінності нейронних мереж для класифікації (А) та генерування ВП (В)

Крок 2. Згорткові шари класифікаційної мережі використовуються для побудови генератора вкладених представлень шляхом заміни кінцевих повнозв'язних шарів на ті, що забезпечуватимуть генерацію ВП (рис. 2.4, В).

До виходів FPN було застосовано активацію ReLU [80] для підкреслення знайдених ознак.

Однак, вихідні дані згорткової частини мережі є високорозмірними, що при їх передачі на шар генерації вкладених представлень призводить до створення надмірної кількості вагів та суттєво збільшує час тренування мережі. Додатково, істотна частина інформації не є значущою та містить «шум».

Одним зі способів вирішення даної проблеми є операція вибору максимуму з групи (max pooling), яка дозволяє суттєво (від 16x16 до 4x4) зменшити розмірність вихідних даних при цьому зберігаючи ключову інформацію з активацій нейронів. Це виконується шляхом проходження вікна («фільтром») фіксованого розміру з певним кроком. Розмір вікна та крок є гіперпараметрами, що впливають на ступінь стиснення даних та збереження інформації. З групи елементів у кожному положенні вікна обирається максимальне значення.

Наприклад, при вікні 2x2 пікселі та кроці у 2, матриця зменшується вдвоє по обидвох осях зі збереженням найбільших елементів (рис. 2.5).



Рисунок 2.5 – Операція групування в максимум

Застосування даної операції до зображень з ідентичними параметрами дає наступний результат. При застосуванні до зображень, операція max pooling зберігає найбільш виражені візуальні елементи, одночасно зменшуючи просторову розмірність. Ключові візуальні характеристики, такі як контури, границі об'єктів та

області високої інтенсивності, зберігаються, тоді як дрібні деталі та текстури згладжуються (рис. 2.6).

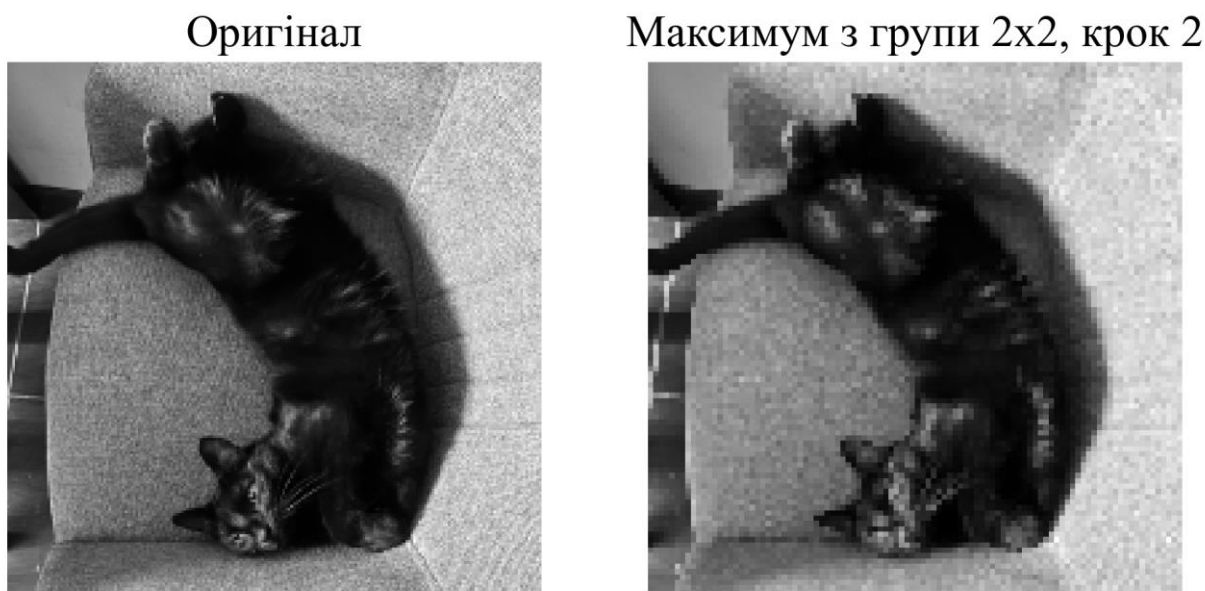


Рисунок 2.6 – Приклад застосування вибору максимуму з групи

Отримані в результаті зменшення розмірності дані були розгорнуті і конкатеновані у вхід для повнозв'язного шару що формує вкладене представлення вхідного зображення.

Одним зі способів тренування нейронної мережі для отримання значущих ВП є вивчення метрики (англ. «metric learning») – навчання з метою побудови векторного простору, у якому схожі вхідні дані знаходяться ближче один до одного, а несхожі – далі. Таким чином, НМ вивчає функцію дистанції між вхідними даними.

Для метричного навчання використано підхід [81], в якому на кожній ітерації формується партія з трійок, що складаються з випадково обраних зображень:

- якірного – зображення, з яким відбувається порівняння (надалі: «якір»);
- позитивного – зображення того ж класу, що і якір (надалі: «позитив»);
- негативного – зображення класу, іншого від класу якоря (надалі: «негатив»).

Під час навчання, функція втрат стимулює наближення у векторному просторі вкладених представлень якіру до позитиву та віддалення від негативу.

У сформованій партії кожен клас представлений однією трійкою, у якій цей клас є якорем. Таким чином, кожна ітерація тренування містить стільки трійок, скільки класів у наборі даних. Однак, при застосуванні до наявних даних такий підхід призвів до слабких результатів у проведених експериментах через занадто малі розміри партій. Таким чином, було реалізовано можливість збільшення кількості даних, що використовуються в окремій тренувальній ітерації. Для цього налаштування було введено гіперпараметр `batch_class_repeats` який визначає скільки разів у партії кожен клас використовується як якор у трійці.

На кожній ітерації навчання за допомогою НМ обчислюються векторні представлення для кожного зображення якоря, позитивного та негативного, і за допомогою них обчислює косинусні подібності (d_{cos}) у парах якор-позитив та якор-негатив (2.1):

$$d_{cos} = \frac{A \cdot B}{\|A\| \|B\|}, \quad (2.1)$$

– де A і B – це векторні представлення порівнюваних зображень. Обчислені подібності (d_{cos}) масштабуються вниз (d_{scaled}) з коефіцієнтом масштабування у $k=0.2$ (2.2):

$$d_{scaled} = d_{cos} * k. \quad (2.2)$$

Відмасштабовані подібності використовуються функцією втрат (`loss`) для зближення векторів пар якор-позитив та віддалення всіх інших. У [81] в якості функції втрат використовується розріджена категоріальна перехресна ентропія (`sparse categorical cross-entropy`) [82]. В результаті експериментів, на ВП отриманих таким чином не вдалось отримати прийнятну точність класифікації, тому було використано наступну трійчасту функцію втрат [83] (2.3):

$$loss = \max(P - N + m, 0), \quad (2.3)$$

– де P і N представляють подібності (2.2) якоря до позитивного та негативного, а m – відстань, на яку вектори повинні бути відокремлені (довільне число більше нуля). Вибір трійчастої функції втрат зумовлено явною орієнтацією на відділення вкладених представлень різних класів одне від одного.

Крок 3. Для виконання класифікації на основі ВП було натреновано інші алгоритми машинного навчання. Для цього, вкладені представлення генеруються для усіх екземплярів тренувальної та тестувальної частин датасету. На основі цих векторів та пов'язаних з ними класів можуть бути навчені довільні класифікаційні алгоритми машинного навчання, наприклад, на основі k -найближчих сусідів чи випадкового лісу.

Таким чином, класифікація на основі ВП містить етапи:

- створення вкладеного представлення зображення за допомогою нейронної мережі;
- класифікація отриманого ВП алгоритмом машинного навчання.

Виконання класифікації за допомогою даного процесу дозволяє уникнути необхідності зміни топології та перетренування нейронної мережі при додаванні нових класів.

2.3.2 Аналіз ефективності класифікаторів на основі вкладених представлень

Ефективність використання вкладених представлень для задач класифікації залежить від двох умов:

1. можливість класифікації на основі ВП досягати точності, співставної з класифікаційною нейронною мережею;
2. розширення нейронної мережі генерації ВП та алгоритмів класифікації за ВП для підтримки нового класу не призводить до істотної втрати точності.

Для перевірки цих умов використано підмножини двох датасетів: MNIST [84] та Fashion-MNIST (далі – FMNIST) [85] (табл. 2.1). Ці набори даних було обрано через їх стандартизованість та широке визнання в науковій спільноті.. Додатково, дані

набори мають чітко визначену структуру класів та контрольовану складність, що дозволяє ізолювати та оцінити ефективність запропонованого методу, а не особливості даних.

Таблиця 2.1 – Використані набори даних

| | MNIST | FMNIST |
|--|-----------------------|----------------------------------|
| Розмір зображення | 28x28 | 28x28 |
| Колірна гама | Градації сірого | Градації сірого |
| Вміст зображень | Рукописні цифри | Одяг |
| Розмір набору даних (тренувальний/тестувальний) | 60000/10000 | 60000/10000 |
| 3-класова підмножина (тренувальний/тестувальний) | 18623/3147 | 18000/3000 |
| 4-класова підмножина (тренувальний/тестувальний) | 24754/4157 | 24000/4000 |
| Використані класи | «0», «1», «2», та «3» | Футболка, штани, пуловер, плаття |

Умова можливості класифікації за ВП перевірялась за допомогою трьох класів з кожного датасету. Умова розширюваності перевірялась додаванням четвертого класу з відповідних датасетів.

Датасет-1 (MNIST). Першим датасетом є стандартний набір даних з класифікації зображень, MNIST, який є підмножиною більшого набору Національного Інституту Стандартів та Технології (США). MNIST складається з зображень рукописних цифр від «0» до «9» з двох спеціальних баз даних NIST: номеру 1 (Special Dataset 1, SD-1) та номеру 3 (Special Dataset 3, SD-3). Початково, NIST пропонував використання набору SD-3 для тренування та SD-1 для оцінювання. Однак, датасети мають принципово різне походження та, відтак, зображення мають суттєво різний ступінь розбірливості написаних цифр. Для формування обґрунтованих висновків з навчальних експериментів необхідно, щоб результат не залежав від вибору тренувального та тестового наборів. Тому, тренувальну частину MNIST створено з 30000 зображень з SD-1 та 30000 зображень з SD-3. Так само, тестувальну частину було набрано з 5000 зображень SD-1 та 5000 зображень SD-3. Тренувальна частина складається з чисел, написаних 250 авторами. Тестувальна

частина не включає зображень, написаних авторами з тестувальної частини. Приклад використаних зображень з даного датасету наведено на рис. 2.7.

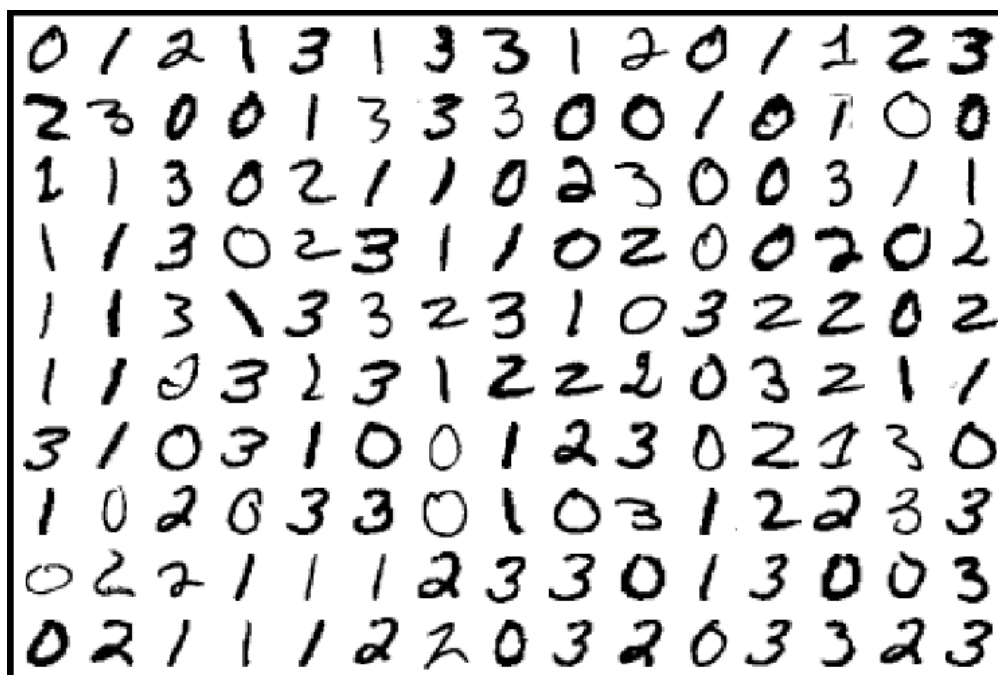


Рисунок 2.7 – Фрагмент набору даних MNIST (колір інвертовано)

Перед включенням в MNIST авторами датасету перед формуванням оригінальні чорно-білі (двокольорові, без відтінків сірого) зображення з баз даних NIST було попередньо оброблено шляхом нормалізації до вміщення в бокс 20*20 пікселів з збереженням відношення сторін, що дало відтінки сірого через вбудоване згладжування алгоритму нормалізації. Для отриманих зображень було обчислено «центр тяжіння» пікселів, та позиціоновано у зображенні 28*28 пікселів, укладаючи «центр тяжіння» цифри у центр зображення-результату.

Хоча MNIST часто використовується як базовий датасет для перевірки нових концепцій, для сучасних методів він є занадто простим. Навіть базові алгоритми машинного навчання легко досягають високих результатів на цьому наборі даних.

Датасет-2 (FMNIST). Другим датасетом, обраним для перевірки гіпотези щодо розширюваності є FMNIST, запропонований дослідниками з команди Zalando, німецької компанії з онлайн-торгівлі одягом.

Щодо формату, вигляду та кількостей даних, а також кількості класів, FMNIST є абсолютно ідентичним до MNIST і є його еквівалентною заміною, та не вимагає жодних змін у кодї окрім зміни джерела даних.

Приклади використаних зображень з даного датасету наведено на рис. 2.8.

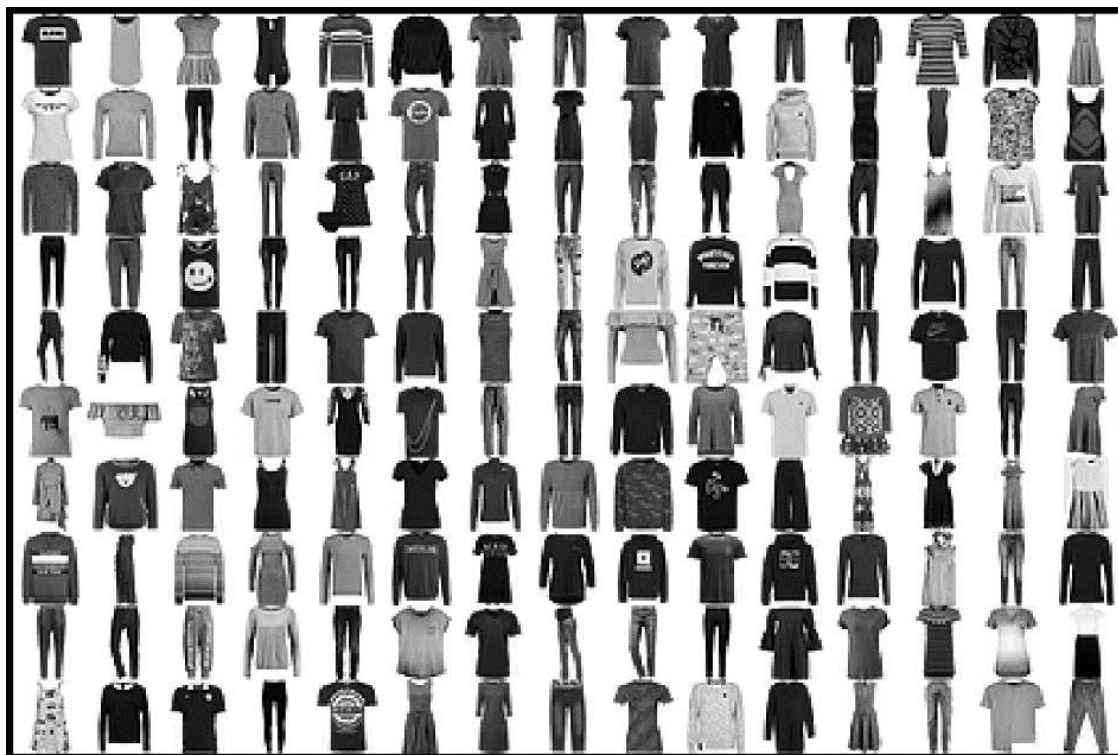


Рисунок 2.8 – Фрагмент набору даних Fashion-MNIST (колір інвертовано)

FMNIST складається з зображень елементів одягу з каталогу Zalando, які авторами датасету було відмасштабовано до розміру 28*28 для сумісності з MNIST у розмірності зображення, підвищено чіткість за допомогою проходу оператором Гауса, та переведено у сумісний з MNIST чорно-білий формат з глибиною відтінків у 8 біт [85].

Для означення класу кожного зображення було використано категорію продукту, вже наявну в базі даних. По 6000 зображень кожного з 10 класів складають тренувальну частину датасету, додаткові 1000 зображень кожного класу складають тестувальну частину.

FMNIST включає в себе наступний набір класів:

- 1) T-shirt/Top (футболка, топ) (мітка класу «0»);

- 2) Trouser (штани) (мітка класу «1»);
- 3) Pullover (свєтєр) (мітка класу «2»);
- 4) Dress (плаття) (мітка класу «3»);
- 5) Coat (куртка) (мітка класу «4»);
- 6) Sandals (босоніжки) (мітка класу «5»);
- 7) Shirt (сорочка) (мітка класу «6»);
- 8) Sneaker (крєсївок) (мітка класу «7»);
- 9) Bag (сумка) (мітка класу «8»);
- 10) Ankle boots (чєревики) (мітка класу «9»);

з яких для перевірки класифікатора було використано перші чотири.

Послідовні згорткові операції суттєво змінюють розмірність даних (так, ResNet-частина нейронної мережі, створеної в розділі 2.2 перетворює зображення $512 \times 512 \times 1$ на $16 \times 16 \times 256$ протягом 5 шарів). Таким чином, подача занадто малих зображень призводить до того, що у нейронної мережі можуть «закінчитись» виміри для проведення згорткових операцій, що призводить до неможливості обчислення останніх шарів нейронної мережі. Для уникнення даної проблеми вхідні зображення було відмасштабовано з розміру 28×28 до 128×128 .

Для перевірки гіпотези про розширюваність без істотної втрати точності розроблено наступний алгоритм:

1. навчимо класифікаційну нейронну мережу;
2. використаємо її ваги для тренування НМ, що генерує вкладені представлення;
3. згенеруємо ВП для кожного елемєнту датасету;
4. за допомогою датасету ВП натренуємо інші алгоритми машинного навчання.

Класифікаційну нейронну мережу для датасету MNIST було навчено на 3 класах до моменту, коли мережа перестанє набувати точність (що зайняло 4 епохи), та отримано точність у 99.8%, використовуючи оптимізатор Nadam [86] з темпом навчання $lr=1e-4$ (визначає, наскільки сильно оновлюються ваги мережі після кожної ітерації) та категоріальною крос-ентропією як функцією втрат. Ідентичну мережу для FMNIST було навчено протягом 8 епох, з отриманням точності у 99.8%.

Для створення вкладених представлень MNIST та FMNIST було побудовано НМ на основі ваг згорткових частин класифікаційних мереж. Мережі генерування ВП було натреновано за підходом, описаним у 2.3.1 з наступними гіперпараметрами:

- `batch_class_repeats=15` кількістю повторень класу у партії;
- `k=0.2` коефіцієнтом масштабування подібностей;
- `lr=1e-3` темпом навчання.

за допомогою оптимізатора Nadam, Точкою припинення навчання було емпірично обрано досягнення значення функції втрат менше ніж у 5000. Цей поріг був вибраний, оскільки в попередніх експериментах було помічено, що досягнення цього значення функції втрат дає гарні результати за прийнятний час. Під час навчання ваги запозичених згорткових шарів було використано як сталі (`frozen`) для використання знань, набутих під час їх навчання на попередньому етапі.

За допомогою навчених мереж було побудовано вкладені представлення усіх зображень датасетів MNIST та FMNIST. На основі отриманих векторів було навчено та оцінено серію простіших алгоритмів класифікації машинного навчання.

Для перевірки гіпотези про розширюваність до навчальних даних мереж, що генерують вкладені представлення було додано четвертий клас. Мережі отримали додаткове навчання на розширеному наборі даних. Оскільки поточний підхід не гарантує сумісності вкладених представлень між початковою та розширеною мережею, для зображень чотирьох класів MNIST та FMNIST було повторно згенеровано ВП. За допомогою нових ВП було натреновано класифікатори на основі:

- методу опорних векторів [87];
- `k`-найближчих сусідів [88];
- випадкового лісу [89].

Приклад отриманих 64-вимірних векторів для зображень чотирьох класів обидвох наборів даних наведено на рис. 2.9 та 2.10.

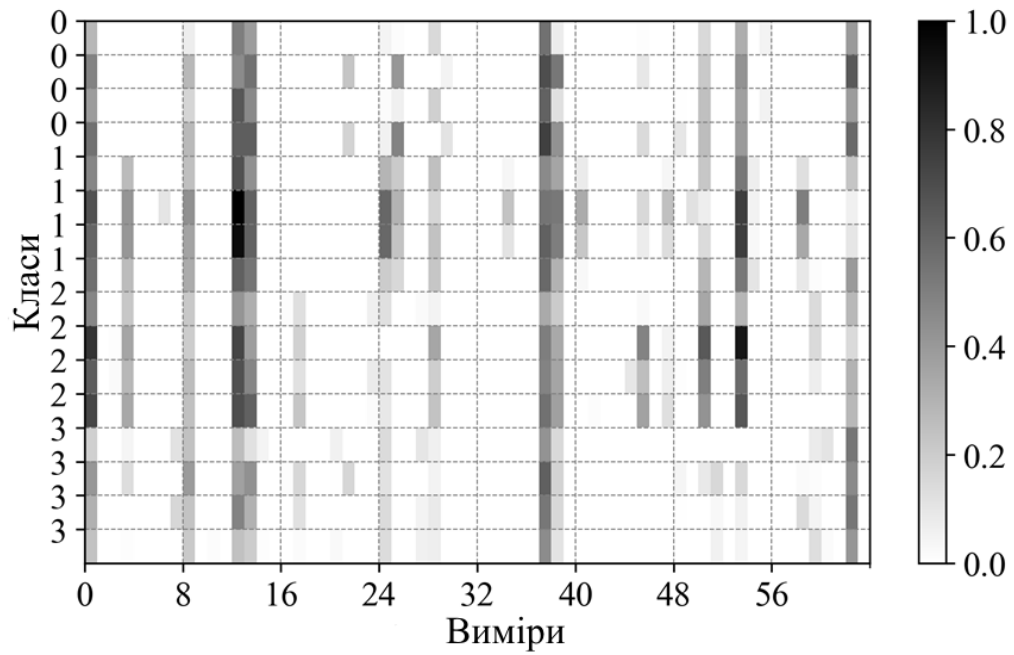


Рисунок 2.9 – Приклади вкладених представлень зображень MNIST

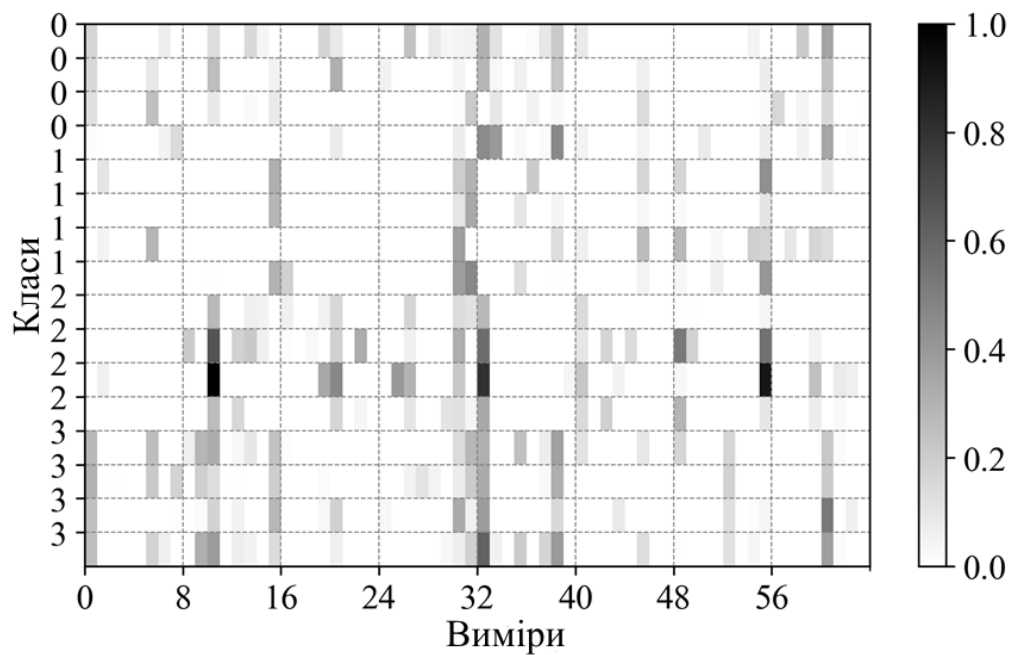


Рисунок 2.10 – Приклади вкладених представлень зображень FMNIST

На рис. 2.10:

- клас 0 позначає T-shirt/Торп (футболка, топ);
- клас 1 позначає Trousers (брюки);
- клас 2 позначає Pullover (свeтp);
- клас 3 позначає Dress (плаття).

Кожен рядок рисунка містить вкладене представлення одного зображення. Чотири послідовних рядки містять ВП одного класу (позначка мітки класу зліва). Вміст векторів було нормалізовано від 0 до 1.

З наведених зображень можна зробити спостереження, що ВП зображень однакових класів дійсно відрізняються один від одного помітно менше ніж зображення різних класів. Це свідчить про успішне навчання нейронної мережі виділяти характерні ознаки кожного класу. Так, помітними на рис. 2.9 (MNIST), окрім інших відмінностей, є наступні:

- класу «0» октетом 0-8;
- класу «1» октетом 24-32;
- класу «2» октетом 40-48;
- класу «3» октетом 48-56.

Дані FMNIST (одяг) містять більше різномайття, деталей та нюансів ніж MNIST (рукописні числа), що відображається і в утворених вкладених представленнях (рис N). Відтак, пошук особливостей ВП класів даного датасету вручну є значно складнішим.

Для перевірки гіпотез про точність, класифікаційні нейронні мережі було оцінено за поведінкою на трьох класах тестових наборів MNIST та FMNIST. Класифікатори на основі вкладених представлень були оцінені на ВП, згенерованих з відповідних тестових наборів (3- та 4-класові частини MNIST та FMNIST).

Оскільки розглядається лише порівняння двох методів класифікації між собою на теоретичних даних, достатнім є порівняння лише точностей. Класифікаційні нейронні мережі з трьома класами досягли точності 0.998 як для MNIST, так і для FMNIST. Це демонструє високу ефективність обох підходів для базового випадку з трьома класами. Точності 3- та 4-класових класифікаторів навчених на ВП у реалізації бібліотеки Scikit-Learn [90] зі стандартними для них гіперпараметрами наведені у Таблиці 2.2.

Таблиця 2.2 – Точності для класифікаторів на основі вкладених представлень

| Класифікатор | 3-класова точність, MNIST/FMNIST | 4-класова точність, MNIST/FMNIST |
|----------------------------|-------------------------------------|-------------------------------------|
| Метод опорних векторів | 0.997/0.977 | 0.987/0.943 |
| k-найближчих сусідів | 0.999/0.978 | 0.988/0.932 |
| Випадкового лісу | 0.997/0.979 | 0.988/0.944 |
| Класифікаційна НМ (без ВП) | 0.998/0.998 | Не перевірялось |

Таким чином, хоча ці результати можна покращити, змінивши алгоритм навчання для мережі векторних представлень і підібравши гіперпараметри класифікаторів на основі ВП, результати, наведені в Таблиці 2.2, дійсно доводять, що такий підхід здатний забезпечити створення систем класифікації, які можна розширювати без істотної втрати точності, що зробило б систему непридатною для використання.

Наведені результати показують, що розроблений метод генерації вкладених представлень та класифікації за ними є застосовним до задач класифікації та дозволяє розширення новими класами без зміни топології нейронної мережі.

2.3.3 Аналіз часових характеристик використання вкладених представлень для класифікації

Для оцінки ефективності використання вкладених представлень (ВП) у класифікації необхідно порівняти накладні витрати часу на розробку та донавчання класифікаторів на основі нейронних мереж і тих, що працюють із ВП. У випадку з ВП цей процес включає не лише навчання та оновлення класифікаторів, але й тренування нейронної мережі для генерації ВП.

Розглянемо витрати часу на початкове тренування традиційної класифікаційної мережі (2.4):

$$t_{\text{трад.повн.}} = t_{\text{ф.д.}} + t_{\text{реал.}} + t_{\text{трен.}}, \quad (2.4)$$

де:

- $t_{\text{трад.повн.}}$ є повним часом на отримання традиційної класифікаційної мережі;
- $t_{\text{ф.д.}}$ є часом на формування та попередню обробку набору даних;

- $t_{\text{реал.}}$ є часом на реалізацію коду нейронної мережі;
- $t_{\text{трен.}}$ є часом на тренування нейронної мережі.

У порівнянні, при використанні підходу з вкладеними представленнями з'являються додаткові витрати часу (2.5):

$$t_{\text{вкл.повн.}} = t_{\text{ф.д.}} + t_{\text{реал.}} + t_{\text{трен.}} + t_{\text{реал.к.}} + t_{\text{трен.к.}} \quad (2.5)$$

де:

- $t_{\text{вкл.повн}}$ є повним часом на отримання класифікаційного методу на основі вкладених представлень;
- $t_{\text{реал.к.}}$ є часом на реалізацію коду класифікатора (-ів) на основі вкладених представлень;
- $t_{\text{трен.к.}}$ є часом на тренування класифікатора (-ів) на основі вкладених представлень (є значно меншим за тренування нейронної мережі, та є таким, яким можна знехтувати).

Розглянемо витрати часу на додавання класу до традиційної класифікаційної мережі (2.6):

$$t_{\text{трад.р.повн.}} = t_{\text{д.д.}} + t_{\text{мод.}} + t_{\text{трен.}} \quad (2.6)$$

де:

- $t_{\text{трад.р.повн}}$ є повним часом на отримання традиційної класифікаційної мережі розширеної підтримкою нового класу;
- $t_{\text{д.д.}}$ є часом на додавання даних нового класу до наявного датасету;
- $t_{\text{мод.}}$ є часом на модифікацію коду нейронної мережі;
- $t_{\text{трен.}}$ є часом на тренування нейронної мережі.

Натомість у випадку використання вкладених представлень модифікація коду не є необхідною, що призводить до таких витрат часу (2.7):

$$t_{\text{вкл.р.повн.}} = t_{\text{д.д.}} + t_{\text{дод.трен.}} + t_{\text{трен.кл.}} \quad (2.7)$$

де:

- $t_{\text{вкл.р.повн.}}$ є повним часом на отримання класифікаційного методу на основі вкладених представлень, розширеного новим класом;
- $t_{\text{дод.трен.}}$ є часом на додаткове тренування нейронної мережі.
- $t_{\text{трен.кл.}}$ є часом на перетренування класифікаторів на отриманих вкладених представленнях.

За умови, що для практичних задач час дотренування НМ є меншим за час на повне перетренування (2.8):

$$t_{\text{дод.трен.}} \leq t_{\text{трен.}} \quad (2.8)$$

та час на перетренування класифікаторів за ВП є значно меншим за час, необхідний для модифікації топології нейронної мережі (2.9):

$$t_{\text{трен.кл.}} \ll t_{\text{мод.}} \quad (2.9)$$

загальний час додавання нового класу до класифікатора на основі вкладених представлень буде істотно менший (2.10-2.11):

$$\begin{aligned} \Delta_t &= t_{\text{трад.р.повн.}} - t_{\text{вкл.р.повн.}} = \\ &= (t_{\text{д.д.}} + t_{\text{мод.}} + t_{\text{трен.}}) - (t_{\text{д.д.}} + t_{\text{дод.трен.}} + t_{\text{трен.кл.}}) = \\ &= (t_{\text{трен.}} - t_{\text{дод.трен.}}) + (t_{\text{мод.}} - t_{\text{трен.кл.}}) \geq 0 \end{aligned} \quad (2.10)$$

$$\Delta_t \geq 0 \text{ при } \begin{cases} t_{\text{дод.трен.}} \leq t_{\text{трен.}} \\ t_{\text{трен.кл.}} \ll t_{\text{мод.}} \end{cases} \quad (2.11)$$

Таким чином, оскільки отримання нового класифікатора на основі вкладених представлень вимагає втручання людини тільки на етапі додавання даних, інші етапи легко можуть бути автоматизовані, що дозволяє значно пришвидшити дотренування для підтримки нових класів за виконання умов (2.8) та (2.9).

Для простих даних, таких як MNIST, витрати часу на реалізацію та тренування розглянутого методу не виправдовують означених переваг (2.8- 2.10 не виконуються), тому тренування звичайних класифікаційних нейронних мереж є порівняно швидшим для таких задач. Однак, для складних задач, що включають більші зображення з більшим різноманіттям форм та текстур, класифікація на основі ВП є ефективною, оскільки дозволяє дотренування нейронної мережі без зміни топології, що може бути важливим у випадках де якнайшвидше додавання нових класів є пріоритетним. Додатково, запропонований метод дає додаткові можливості у вигляді використання отриманих вкладених представлень у векторних базах даних для пошуку схожих зображень.

2.4 Мультикласова класифікація знімків КТ з використанням вкладених представлень

При мультикласовій класифікації знімків КТ відмінності у порівнянні з використанням класифікаційної нейронної мережі полягатимуть у наступному:

- перед виконанням класифікації необхідною буде генерація вкладеного представлення зрізу КТ;
- класифікація виконуватиметься за допомогою інших алгоритмів навчання, вибір яких має бути обґрунтований експериментальним дослідженням ефективності їх роботи з ВП.

2.4.1 Побудова вкладених представлень знімків КТ

Для утворення генератора вкладених представлень за основу було взято згорткові шари та їх ваги з класифікаційної нейронної мережі, натренованої у розділі 2.2. За запропонованим методом, до виходів FPN було застосовано активацію ReLU, зменшено їх розмірність за допомогою вибору максимуму з групи (max pooling), та подано на вхід до повнозв'язного шару для створення ВП довжиною у 128 вимірів. Відтак, загальна топологія нейронної мережі для генерування ВП зрізів знімка КТ виглядає наступним чином (рис. 2.11).

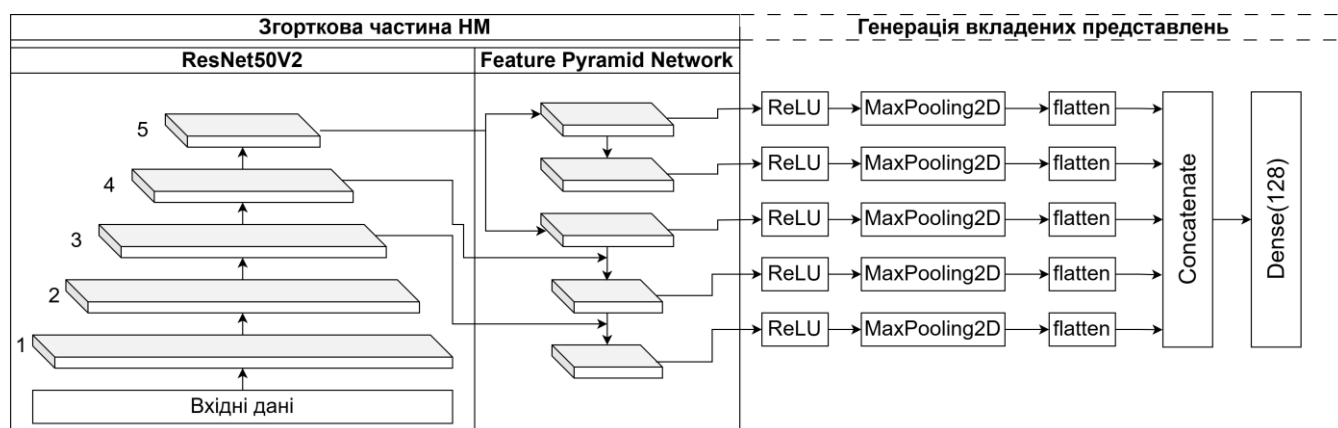


Рисунок 2.11 – Нейронна мережа для генерування вкладених представлень

Згорткові шари утвореного генератора ВП було ініційовано вагами згорткових шарів з класифікаційної нейронної мережі та використано як сталі (frozen). Таким чином, вивчені цими шарами у класифікаційній мережі ознаки використовуються для створення вкладених представлень, але ваги ЗНМ не змінюються.

Згідно з запропонованим методом, нейронну мережу було натреновано за допомогою порівняння вкладених представлень зображень у трійках якір-позитив-негатив. В партії було включено по 6 трійок на клас (COVID-19, позагоспітальна пневмонія, здорові легені). Зображення у трійках обирались випадковим чином. Для якісного узагальнення вивчених ознак до обраних зображень застосовується ряд випадкових трансформацій:

- поворот зображення;
- зміщення по вертикалі та горизонталі;
- масштабування;
- дзеркальні відображення;
- зсув (перетворення прямокутника на паралелограм);
- зміну яскравості.

Використання даних трансформацій входить в техніку аугментації даних, та дозволяє, окрім узагальнення, запобігти перенавчанню моделі, а також істотно розширяє набір наявних для навчання якісних даних, оскільки навіть сусідні зрізи КТ

або один і той самий зріз будуть подані мережі у істотно видозміненому вигляді через застосовані трансформації.

Для обчислення втрат було використано трійчасту функцію (2.3).

Для тестування було поставлено обчислювальний експеримент, у якому нейронну мережу було натреновано протягом 4300 наборів (18 трійок у наборі, 6 трійок на клас). Після тренування векторизатора, повний датасет (включаючи і тренувальні, і тестувальні дані) було перетворено на вкладені представлення (приклади у нормалізованому вигляді наведено в «Додаток Б»), за якими було навчено низку класифікаційних алгоритмів машинного навчання.

Таким чином, було отримано генератор вкладених представлень знімків КТ та ряд класифікаторів на їх основі. Класифікацію захворювання за знімком виконується наступним чином:

1. створюється ВП за допомогою нейронної мережі;
2. відбувається класифікація ВП обраним алгоритмом машинного навчання.

Для отриманих класифікаторів було оцінено ефективність за допомогою низки метрик на тестовій частині датасету. Серед натренованих класифікаційних алгоритмів найкращу точність (98.7%) вдалось досягти за допомогою методу k-найближчих сусідів. Таким чином доведено, що запропонований метод побудови вкладених представлень дозволяє використання у задачах аналізу знімків КТ для діагностики захворювань легень.

2.4.2 Застосування алгоритмів машинного навчання до класифікації вкладених представлень знімків КТ

Розглянемо детальніше метрики якості класифікації, досягнуті у результаті обчислювального експерименту з класифікації вкладених представлень зображень КТ. З метою порівняння ефективності різних алгоритмів на ВП знімків КТ було натреновано наступні класифікатори:

- метод опорних векторів (Support Vector Classifier) [87];
- метод k-найближчих сусідів (KNeighbors Classifier) [88];
- метод випадкового лісу (Random Forest Classifier) [89];

- ансамблевий класифікатор на основі голосування [91], що використовує попередні методи;
- AdaBoost [92];
- Гаусівський процес (ГП) (Gaussian Process Classifier) [93];
- Аналіз квадратичних дискримінант (АКД) (Quadratic Discriminant Analysis) [94].

Досліджені алгоритми було використано у реалізації, наявній у бібліотеці алгоритмів машинного навчання Scikit-learn [90]. Кожним з алгоритмів було проведено класифікацію вкладених представлень тестувальної частини набору даних. На основі порівняння отриманих та вказаних в датасеті класів засобами Scikit-learn було обчислено ряд метрик:

- точність (2.12);

$$\text{ТОЧНІСТЬ} = \frac{\text{Коректно класифіковані}}{\text{Усі класифіковані}} \quad (2.12)$$

- F1-міра (2.13);

$$F1 = 2 \times \frac{\text{влучність} \times \text{повнота}}{\text{влучність} + \text{повнота}} \quad (2.13)$$

- влучність (2.14);

$$\text{влучність} = \frac{\text{ІП}}{\text{ІП} + \text{ХП}} \quad (2.14)$$

- повнота (2.15).

$$\text{повнота} = \frac{\text{ІП}}{\text{ІП} + \text{ХН}} \quad (2.15)$$

– де ІП – істинно позитивні результати, ІН – істинно негативні, ХП – хибно позитивні, ХН – хибно негативні.

Набір метрик обґрунтований необхідністю у детальнішому дослідженні поведінки окремих алгоритмів. Так, влучність (2.13) дозволяє оцінити якість класифікації з огляду на кількість хибно позитивних класифікацій, а повнота (2.14) –

хибно негативних. У свою чергу, F1-міра (2.12), як гармонійне середнє між влучністю та повнотою, є особливо важливою, оскільки забезпечує збалансовану оцінку якості класифікації з урахуванням ризиків як хибно позитивних, так і хибно негативних результатів. Отримані значення наведено у таблиці 2.3.

Таблиця 2.3 – Метрики якості класифікації на основі ВП

| Назва | Точність | F1-міра | Влучність | Повнота |
|----------------------------------|----------|---------|-----------|---------|
| Метод опорних векторів | 0.920 | 0.920 | 0.921 | 0.920 |
| Метод К-найближчих сусідів | 0.987 | 0.987 | 0.987 | 0.987 |
| Метод випадкового лісу | 0.960 | 0.960 | 0.961 | 0.960 |
| Ансамблевий класифікатор | 0.978 | 0.978 | 0.978 | 0.978 |
| AdaBoost | 0.830 | 0.830 | 0.843 | 0.830 |
| Гаусівський процес | 0.515 | 0.383 | 0.653 | 0.515 |
| Аналіз квадратичних дискримінант | 0.766 | 0.753 | 0.816 | 0.766 |

F1-міра, влучність та повнота є орієнтованими на бінарну класифікацію, оскільки обчислюються з матриці невідповідності (що містить кількість істинно позитивних, істинно негативних, хибно позитивних, хибно негативних результатів). Відтак, у випадку трьох класів необхідно обчислювати окремі значення для кожного з класів, та приведення їх до одного показника шляхом усереднення. При обчисленні метрик відносно результатів даного експерименту було використано зважене середнє \bar{x} (2.16).

$$\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}, \quad (2.16)$$

– де \bar{x} – зважене середнє, n – кількість елементів у наборі, x_i – елемент набору та w_i – вага елементу x_i . Для обчислення загальних метрик елементами є окремі метрики для кожного класу вагою є кількість істинних елементів цього класу у наборі даних.

Результати експериментів при вирішенні задачі класифікації знімків КТ з використанням ВП, показують що серед розглянутих алгоритмів кращі результати показали методи:

- k-найближчих;
- випадкового лісу;

– ансамблевий метод.

Дані методи є універсальними. Однак, гірші результати інших методів вимагають більш детального аналізу (табл. 2.4). Матриці для кожного алгоритму окремо наведено у «Додаток А».

Таблиця 2.4 – Порівняння матриць невідповідності

| Назва | Передбачений клас / істинний клас * | | | | | | | | |
|----------------------------------|-------------------------------------|-----|-----|-----|------|------|-----|-----|------|
| | П/П | К/П | З/П | П/К | К/К | З/К | П/З | К/З | З/З |
| Метод опорних векторів | 529 | 19 | 25 | 7 | 2000 | 246 | 11 | 125 | 2469 |
| Метод К-найближчих сусідів | 567 | 5 | 1 | 5 | 2209 | 39 | 4 | 15 | 2586 |
| Метод випадкового лісу | 517 | 31 | 25 | 0 | 2127 | 126 | 0 | 31 | 2574 |
| Ансамблевий класифікатор | 553 | 9 | 11 | 0 | 2167 | 86 | 3 | 10 | 2592 |
| AdaBoost | 525 | 26 | 22 | 22 | 2045 | 186 | 16 | 649 | 1940 |
| Гаусівський процес | 0 | 0 | 573 | 0 | 192 | 2061 | 0 | 0 | 2605 |
| Аналіз квадратичних дискримінант | 414 | 9 | 150 | 207 | 1192 | 854 | 32 | 18 | 2555 |

* де:

- К – COVID-19;
- П – позагоспітальна пневмонія;
- З – здорові легені.

Найгірші результати було отримано з AdaBoost, Гаусівським процесом та аналізом квадратичних дискримінант. AdaBoost є чутливим до шуму та відхилень у даних, та вимагає тонкого налаштування для вивчення закономірностей у складніших даних, що пояснює отримані при дослідженні результати [92]. Гаусівський процес [95] та аналіз квадратичних дискримінант [94] мають низку обмежень, що впливають на їхню універсальність для різних задач класифікації та часто вимагають ручного налаштування для досягнення прийнятних результатів. У даному експерименті процес побудови генерує ВП у 128-вимірному просторі та не може гарантувати виконання даних припущень, отримана поведінка даних алгоритмів є очікуваною.

2.5 Висновки до розділу

У даному розділі наведено результати ряду експериментів, які дозволили встановити наступне:

- доведено можливість розширення бінарного класифікатора знімків КТ новим класом без істотної втрати точності (до 5%);
- розроблено модифікацію існуючої нейронної мережі класифікації для побудови вкладених представлень з класифікацією на основі отриманих ВП за допомогою інших методів машинного навчання (на датасетах MNIST та Fashion-MNIST);
- доведено можливість додавання нових класів у класифікатор на основі ВП без зміни топології НМ та істотної втрати точності (на датасетах MNIST та Fashion-MNIST);
- доведено можливість використання класифікатора на основі ВП для аналізу зрізів знімків КТ (досягнуто точність у 98.7%);

Проведено порівняння алгоритмів машинного навчання з бібліотеки Scikit-Learn для класифікації за вкладеними представленнями та зроблено висновки щодо їх застосовності.

Розроблено математичну модель для оцінки часу на побудову та донавчання класифікатора та наведено умови, що визначають ефективність використання вкладених представлень шляхом врахування складових часу модифікації.

На відміну від існуючих підходів до класифікації знімків КТ, що в основному зосереджуються на підвищенні якості класифікації та не звертають увагу на час впровадження даних рішень, запропонований метод класифікації зображень з використанням ВП дозволяє донавчання нейронної мережі з метою підтримки нових класів без зміни топології, що зменшує час на адаптацію програмного забезпечення аналізу медичних зображень для діагностики. Дана властивість дозволяє пришвидшити донавчання класифікатора як системи з генератора вкладених представлень та класифікаторів, що є важливим для задачі діагностики захворювань, оскільки під час спалаху нового захворювання існує необхідність у якнайшвидшій появі інструментів для його виявлення.

Недоліком використання вкладених представлень для класифікації зображень, зокрема КТ, є ускладнений процес реалізації та початкового тренування, що створює додаткові накладні витрати часу при розробці програмного забезпечення.

На сформованому датасеті знімків КТ визначено основні характеристики запропонованого методу.

3 ПРОГРАМНА СИСТЕМА КЕРУВАННЯ ПОТОКАМИ РОБІТ

Розширення системи аналізу зображень для діагностики новими даними та/або методами вимагає втручання в послідовність виконуваних операцій. Оскільки існуючі системи керування потоками робіт вимагають жорсткого визначення графу потоку робіт, доцільною є розробка методів та систем, що конструюють потоки робіт динамічно з наданих операторів та даних.

До переваг такого підходу відноситься зменшення витрат часу на:

- модифікацію потоків робіт, що виконуються системою (у випадку аналізу зображень достатньо зареєструвати новий метод до існуючих вхідних даних і система автоматично додасть його в потоки робіт);
- можливість інтеграції інших форматів збереження та передачі оброблюваних даних без втручання в код системи. Наприклад, нехай існуючі методи певної системи аналізу зображень орієнтовані на аналіз окремих зрізів знімка КТ. Тоді, при появі методу, що працює з знімками у трьох вимірах, та вимоги підтримувати завантаження тривимірних DICOM-файлів, для підтримки нового методу достатньо оголосити в СКПР новий вид даних та новий оператор (див. нижче). Для зворотної сумісності зі старими методами достатньо додати оператор, що розпаковуватиме тривимірний DICOM у двовимірні зрізи. Варто зауважити, що ці зміни не потребуватимуть змін у кодовій базі самої системи, а лише додавання окремих елементів. Таким чином, ці зміни не призводять до модифікації вхідного коду, що відповідає за комунікацію частин системи.

Для реалізації та перевірки динамічної побудови графу ПР під час його виконання необхідно:

- визначити сутності, з яких складається потік робіт;
- визначити обмеження, що накладаються на алгоритм та дані;
- реалізувати алгоритм побудови потоків робіт;
- реалізувати СКПР на основі алгоритму;

- виконати експерименти з виміру швидкодії алгоритму з потоками різної складності;
- виконати експерименти з виміру швидкодії виконання СКПР окремих кроків потоку робіт.

В прив'язці до задачі аналізу знімків КТ, з урахуванням обсягів даних та вимог щодо часу впровадження системи, окрему увагу слід приділити питанням паралелізму та масштабованості.

3.1 Динамічна побудова графу потоку робіт під час його виконання

Для втілення динамічної побудови потоку робіт з його складників під час виконання обчислень, алгоритм мусить відповідати низці умов:

- 1) алгоритм має враховувати непередбачуваність поведінки визначених користувачем інструментів, тобто має бути доступна можливість повернути усі оголошені виходи оператору, в будь-якій кількості, або не повернути нічого. Така поведінка дозволяє користувачу реалізовувати умовні переходи та обробляти помилки без зупинки усього потоку робіт;
- 2) алгоритм має вичерпно знайти та виконати усі можливі обчислення (комбінації) над даними та операторами. Під даними слід розуміти не лише первинні дані, а й ті, що повернені операторами на попередніх ітераціях потоку робіт.
- 3) додатково, алгоритм повинен передбачити зациклення обчислень та механізми реакції на них та/або їх уникнення. Відтак, при виконанні потоку робіт необхідним є відстеження походження даних-результатів та врахування цього при запуску наступних операторів. Уникнення циклів є критичним для стабільності системи та уникнення ресурсних та часових витрат, пов'язаних з розв'язанням наслідків їх утворення в СКПР.

3.1.1 Складові частини потоку робіт

Для реалізації запропонованого методу необхідною є низка сутностей, композиція яких дозволяє повністю описати певне обчислення з набором вхідних, проміжних та вихідних даних та по суті являє собою нетривіальний спосіб запису спрямованого ациклічного графу.

Такими сутностями є:

- **вид даних (ВД)** (Data Kind) – є семантичною позначкою, та використовується для анотації будь-яких даних що існують в певному обчисленні. Варто зауважити, що запропонований алгоритм містить методи валідації даних за їх типом, тому ВД є просто строковою позначкою. ВД може бути концептуалізованим як тип у розрізненні типу та токена Чарльза Сандерса Пірса [96], [97];
- **оператор** (Operator) – є атомарною операцією над множиною строго визначених входів різних видів даних, та може повернути довільну кількість довільних видів даних (однак, можливі вислідні види даних мають бути оголошені разом з оператором). Однак ні кількість, ні вид повернених даних не є передбачуваними для запропонованого алгоритму;
- **одиниця даних (ОД)** (Data Unit) – є атомарним елементом даних. Якщо види є типами по Пірсу, то одиниці є токенами. Окрема ОД складається з позначки ВД, посилання на файл, що містить описувані дані та списку родоводу – послідовності операторів, обчислення яких її призвело до появи в множині даних обчислення.
- **крок** – є виконанням конкретного оператора з конкретним набором одиниць даних, відповідних входам оператора. Крок є унікальним в обчисленні, та не може повторюватись.

3.1.2 Опис реалізації методу

Порядок роботи алгоритму, що реалізує запропонований метод, полягає у тому, щоб на кожній ітерації віднайти наступні можливі для виконання кроки, враховуючи

набір наявних одиниць даних і наявні оператори та виконати їх для отримання нових ОД.

Для цього, на кожній ітерації алгоритму визначаються оператори, усі з входів яких можуть бути задовільнені наявними одиницями даних. З цього списку підставленням ОД формуються можливі подальші кроки з урахуванням уникнення циклічних обчислень. Для уникнення циклів виконується перевірка наявності оператора, з яким формується крок, у родоводі вхідних одиниць даних кроку. Після відсіву вже виконаних комбінацій «оператор-дані», виконуються віднайдені наступні кроки, вихідні ОД додаються до множини наявних, після чого цикл розпочинається знову. Умовою зупинку циклу є відсутність нових потенційних кроків для виконання.

Наступний псевдокод детально описує поведінку алгоритму. У псевдокодi використовуються наступні позначення:

- O є множиною наявних операторів;
- K є множиною наявних видів даних;
- S є множиною обчислених у поточній ітерації кроків;
- U є множиною усіх одиниць даних, вхідних і вихідних до операторів, у поточному обчисленні.

Algorithm 1: Main loop

Data: O, K, U що задовільняє $\forall u \in U, u.dataKind \in K$ та містить вхідні дані надані користувачем

Result: U що містить результати вичерпних обчислень за допомогою $o \in O$.

start:

$S \leftarrow \emptyset;$

repeat

$N \leftarrow \text{SolveTurn}(O, K, S, U);$

if $N \neq \emptyset$ **then**

$U \leftarrow U \cup \text{ExecuteSteps}(N, U);$

$S \leftarrow S \cup N;$

end

until $N = \emptyset$

end

Function $\text{SolveTurn}(O, K, S, U):$

/ Обираємо оператори, чиї входи задовільнені хоча б одною одиницею з множини наявних даних */*

```

O1 ← GetAvailableOperators(O, U);
opsWithOptions ← відображення o ∈ O з
  SelectOperatorInputs(o, U);
if opsWithOptions = ∅ then
  return ∅;
end
/* Для кожного оператора, обчислити Декартів добуток усіх
доступних одиниць на кожен вхід для утворення наборів вхід-
них даних для створення кроків */
opsWithInputCombinations ← відображення
opsWithSatisfiedInputs з f(o) =
  (operator:o.operator, inputCombinations:
    {i1...in ∈ o.unitsPerInput|i1 × i2 × ... × in});
newSteps ← ∅;
foreach o ∈ opsWithInputCombinations do
  foreach i ∈ o.inputCombinations do
    if
      i не містить повторів одиниць
    then
      append (operator:o, input:i) to
        availableNewSteps;
    end
  end
end
N ← {n ∈ newSteps | n ∉ S};
return N
Function GetAvailableOperators(O, U):
kindsPresent ← видобути u.dataKind з u ∈ U;
return
  {o ∈ O | o.inputDataKinds ⊆ kindsPresent};
Function SelectOperatorInputs(o, U):
matchingDataUnits ← фільтрувати u ∈ U, аби
  u.dataKind ∈ o.inputDataKinds;
unitsPerInput ← пустий Map;
foreach input ∈ o.inputs do
  unitsPerInput[input.name] ←
    пустий List;
  foreach unit ∈ matchingDataUnits do
    if unit.dataKind = input.dataKind
      and ¬o.name ∈ unit.ancestors then
      append unit.id to
        unitsPerInput[input.name];
  end

```

```

end
end
return (operator, unitsPerInput);

```

Розглянемо роботу алгоритму на прикладах (приклади 1-4) та використаємо наступні позначення:

- великі латинські літери без підрядкової позначки (A, B, ...) є видами даних
- великі латинські літери з підрядковими позначками є одиницями даних відповідного виду (A₁, B₁, ...)
- малі латинські літери є операторами;
- запис $f(A, B) \Rightarrow C$ відповідає оператору, що приймає одиницю виду A в першій вхідній позиції та одиницю виду B в другій вхідній позиції, та повертає одиницю виду C.

Приклад 1. Простий потік робіт

Розглянемо базовий потік робіт (рис. 3.1).

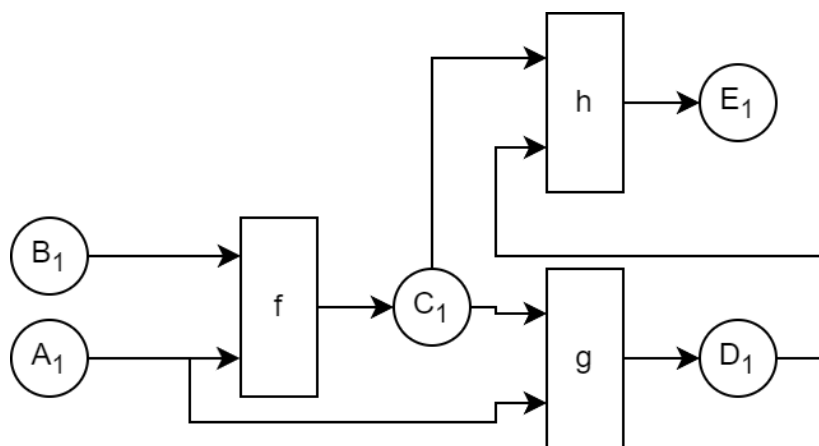


Рисунок 3.1 – Приклад потоку робіт, побудованого та пройденого алгоритмом

У даному потоці робіт використані наступні типи даних: A, B, C, D, E. Шляхом проведення ряду обчислень (рис. 3.1) дані видів A та B за допомогою проміжних ВД C та D дають можливість обчислити E.

Для виконання даних обчислень визначені наступні оператори:

- $f(A, B) \Rightarrow C$;
- $g(A, C) \Rightarrow D$;
- $h(C, D) \Rightarrow E$.

Даний приклад демонструє принцип формування потоку робіт. Однак, такий потік робіт є простим та чітко визначеним, оскільки всі оператори в ньому гарантовано повертають значення фіксованого виду. Такі ПР є примітивними та можуть бути реалізовані за допомогою довільних СКПР.

Приклад 2. «непередбачуваний» оператор

Складнішими (а в деяких СКПР – неможливими) є потоки робіт, в яких оператори мають непередбачувану для алгоритму поведінку. Ця поведінка може бути спричинена як участю випадкових значень у роботі коду оператора, так і високою залежністю результату від вмісту вхідних даних оператора. Для запропонованого алгоритму, обидві причини непередбачуваної поведінки оператора є рівноцінними, оскільки алгоритм не перевіряє вміст даних, які передаються до входів операторів.

Для демонстрації розглянемо приклад, в якому міститиметься непередбачуваний оператор. Для цього видозмінимо список операторів з прикладу 1 наступним чином:

- $f(A, B) \Rightarrow C$;
- $g(A, C) \Rightarrow (D, E)$;
- $h(C, D) \Rightarrow F$;
- $k(B, E) \Rightarrow I$.

У цьому випадку, оператор g може видати як результат своїх обчислень:

- одиницю виду D (одну або декілька);
- одиницю виду E (одну або декілька);
- довільну кількість одиниць обидвох видів;
- не надати результату обчислення.

Потік робіт, що утворюється з таким набором операторів при вхідних ОД A_1 та B_1 представлений на рис. 3.2.

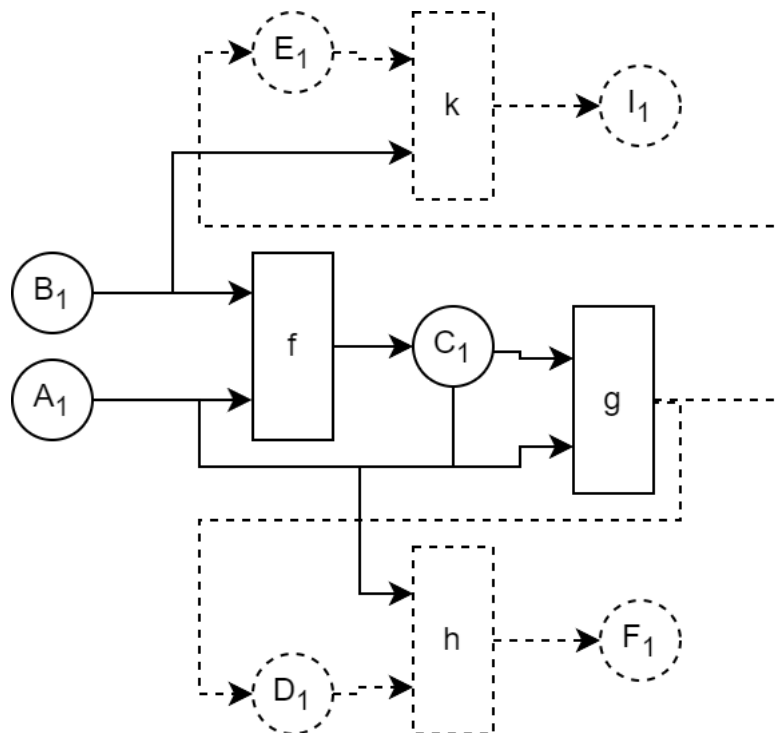


Рисунок 3.2 – Приклад потоку робіт з опціональними результатами оператору

На рисунку пунктирною лінією зображено ОД, поява яких у графі не є гарантованою та залежить від випадку чи вхідних даних (намічені пунктирним оператори можуть бути запуснені за наявності відповідних вхідних даних, або пропущені за їх відсутності).

Відповідно, результатом виконання потоку робіт може бути як лише одиниця даних C_1 (у випадку, якщо оператор g не повернув жодного результату), так і всі п'ять можливих вихідних одиниць – C_1, D_1, E_1, F_1, I_1 , так і три – C_1, D_1, F_1 чи C_1, E_1, I_1 .

Приклад 3. Повернення оператором довільної кількості одиниць даних

Іншим аспектом динамічності побудови потоку робіт є можливість повернення оператором довільної кількості одиниць даних.

Розглянемо приклад, в якому міститиметься оператор, що повертає декілька одиниць одного виду даних. Скористаємось операторами прикладу 1:

- $f(A, B) \Rightarrow C$;
- $g(A, C) \Rightarrow D$;
- $h(C, D) \Rightarrow E$.

Припустимо, що оператор g повертатиме не одну ОД виду D , а довільну їх кількість. Діаграму ПР, що буде обчислено у такому разі при входних ОД A_1 та B_1 представлено на рис. 3.3.

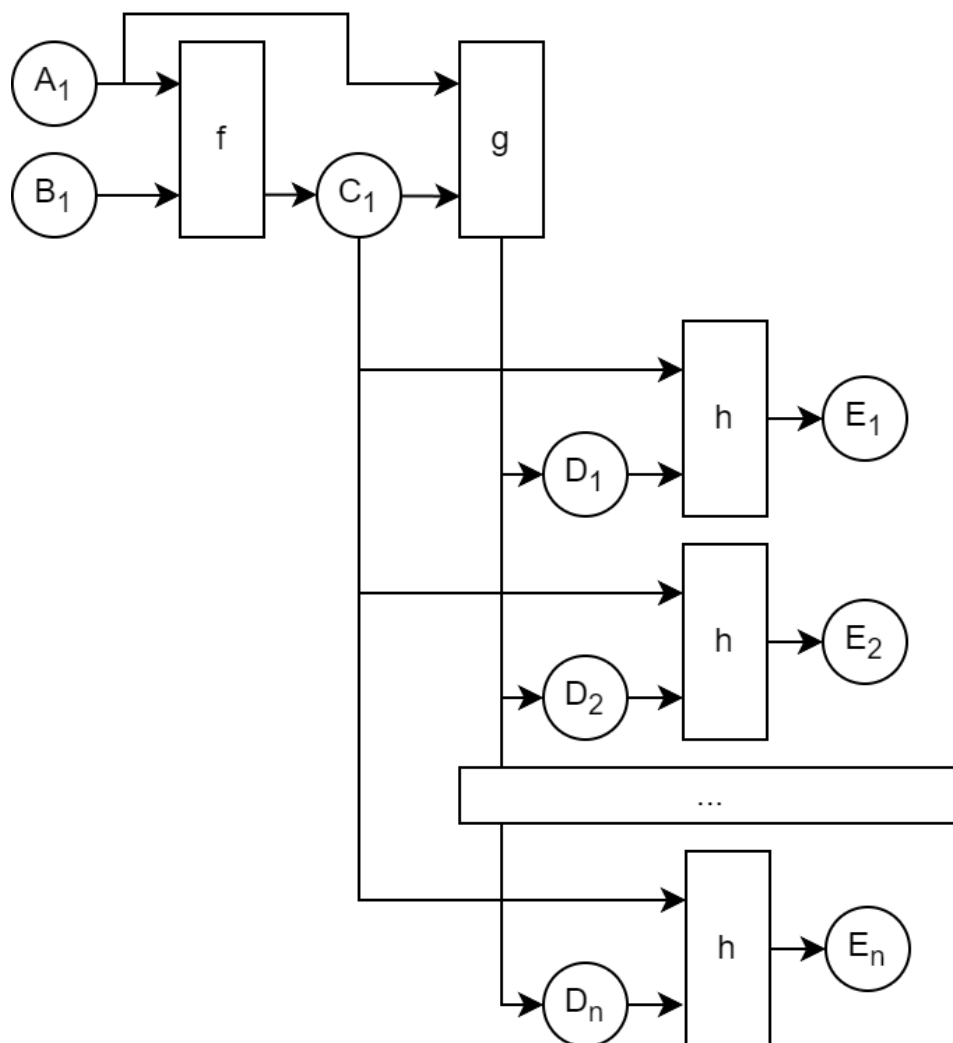


Рисунок 3.3 – Потік робіт з довільною кількістю результатів роботи оператора

Таким чином, будь-який оператор також може повернути довільну кількість ОД певного виду, і, у разі наявності операторів який приймає цей ВД, кожна з цих одиниць буде оброблена відповідними операторами. Оскільки для комбінування наборів входних даних для операторів використовується декартів добуток – у випадку повернення оператором f елементів виду C у кількості m , оператор h буде запущений для кожної пари одиниць видів C та D , продукуючи $m \cdot n$ одиниць виду E . Запропонованим алгоритмом також підтримується довільна кількість входів

оператору, відтак – у потоці робіт може бути довільна кількість одиниць усіх вхідних видів певного оператора, і цей оператор буде запущено для кожної можливої комбінації вхідних даних.

Приклад 4. Уникнення циклічних обчислень

Однією з проблем описаного алгоритму є метод уникання циклічних обчислень. У запропонованому варіанті, уникнення виникнення циклів гарантується лише тим, що операторові не можуть бути подані на вхід дані, які мають його у родоводі. Таким чином, навіть якщо дані проходили через оператор навіть декілька кроків обробки назад, навіть якщо вид даних збігається з вхідними даними оператора, ці дані не будуть подані на вхід до оператора.

Прикладом проблеми в уникненні циклів є гіпотетичний оператор x :

$$x(A, B) \Rightarrow A.$$

Розглянемо потік робіт, що складається лише з оператора x та запускається з одиницями (A_1, B_1) , та повертає одиницю A_2 (рис. 3.4).

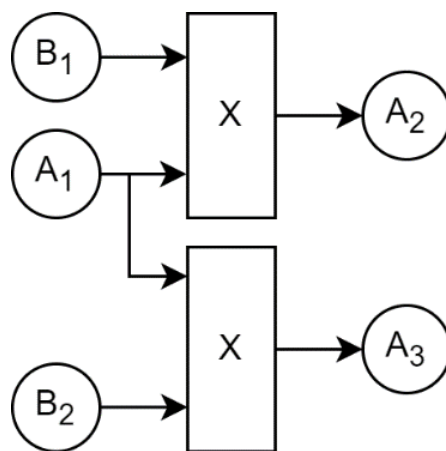


Рисунок 3.4 – Потік робіт з гіпотетичним оператором X , побудований з поточним методом уникнення циклів

При появі (наприклад, породженні іншим оператором) у потоці робіт одиниці B_2 , оператор X повторюється лише з парою входів (A_1, B_2) .

Однак, можемо припустити існування ситуацій, у яких було б також потрібно запустити оператор X також з парою (A_2, B_2) , попри той факт, що A_2 було обчислено за допомогою оператора X . Діаграма потоку робіт у такому випадку зображена на Рис. 3.5.

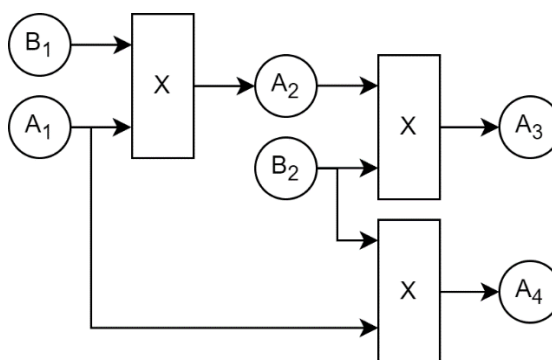


Рисунок 3.5 – Потік робіт з гіпотетичним оператором X, побудований з теоретично можливим іншим методом уникнення циклів

Одним з варіантів вирішення даної проблеми є використання у родоводі ОД не тільки операторів, а і ОД, що подавались до них на вхід. Додатковою перевагою використання родоводу з урахуванням ОД є можливість відстежувати похідними яких даних є певна ОД, що дозволяє використання декількох одиниць одного виду даних як вхідних для потоку робіт без утрати зв'язку між вхідними ОД та результатами обчислень. Даний приклад ілюструє обмеження даного алгоритму, спричинені методом уникання циклічних обчислень.

В даному підрозділі було на практиці показано теоретичні можливості запропонованого алгоритму виконання потоків робіт. При побудові потоків робіт, алгоритм забезпечує уникнення нескінченних циклічних обмежень, підтримує повернення оператором довільної кількості довільних даних. Таким чином, запропонований алгоритм є дієвим для збереження часу та зусиль на створення та модифікацію потоків робіт у обчислювальних системах.

3.2 Програмна система керування потоками робіт з динамічною побудовою графу

Важливим етапом є втілення запропонованого методу у системі керування потоками робіт. Для цієї СКПР описано вимоги до неї, її залежності, процес розгортання, архітектура, технологічні рішення та розроблений функціонал.

Для забезпечення здатності до практичного використання СКПР та її актуальності для вирішення реальних завдань необхідними є такі функціональні можливості:

- визначення видів даних (ВД), що доступні системі;
- визначення операторів, які здійснюють обчислення над наявними ВД;
- створення потоку робіт (run), який представляє одне виконання обчислення на конкретному наборі вхідних даних;
- завантаження даних, їх маркування видом даних та прикріплення до потоку робіт;
- видобування списку операторів що можуть бути виконані в певному потоці робіт та їх параметрів;
- запуск потоку робіт з певним набором параметрів операторів;
- відстеження статусу потоку робіт;
- видобування даних, обчислених в результаті виконання обчислень.

Також, дана СКПР має підтримувати наступні вимоги:

- портативність – СКПР має бути побудована таким чином, аби мінімізувати залежність від конкретних хмарних провайдерів, та має бути готовим до розгортання на будь-якому з них, а також на власних апаратних потужностях, що є необхідним для мінімізації часу розгортання системи;
- масштабованість – СКПР має бути побудована з огляду на непередбачуваність навантаження та відповідне автоматичне горизонтальне масштабування, що є необхідним для мінімізації часу доступу до системи;
- надійність – СКПР має коректно обробляти помилки виконуваних обчислень.

3.2.1 Компоненти системи

Дана система реалізована з використанням наступних фреймворків та технологій:

- середовище виконання Node.js;
- фреймворк NestJS;
- система зберігання файлових даних MinIO;
- система організації кластерних обчислень Kubernetes;
- документно-орієнтована база даних CouchDB.

З урахуванням даних засобів діаграма компонент СКПР виглядає наступним чином (рис. 3.6).

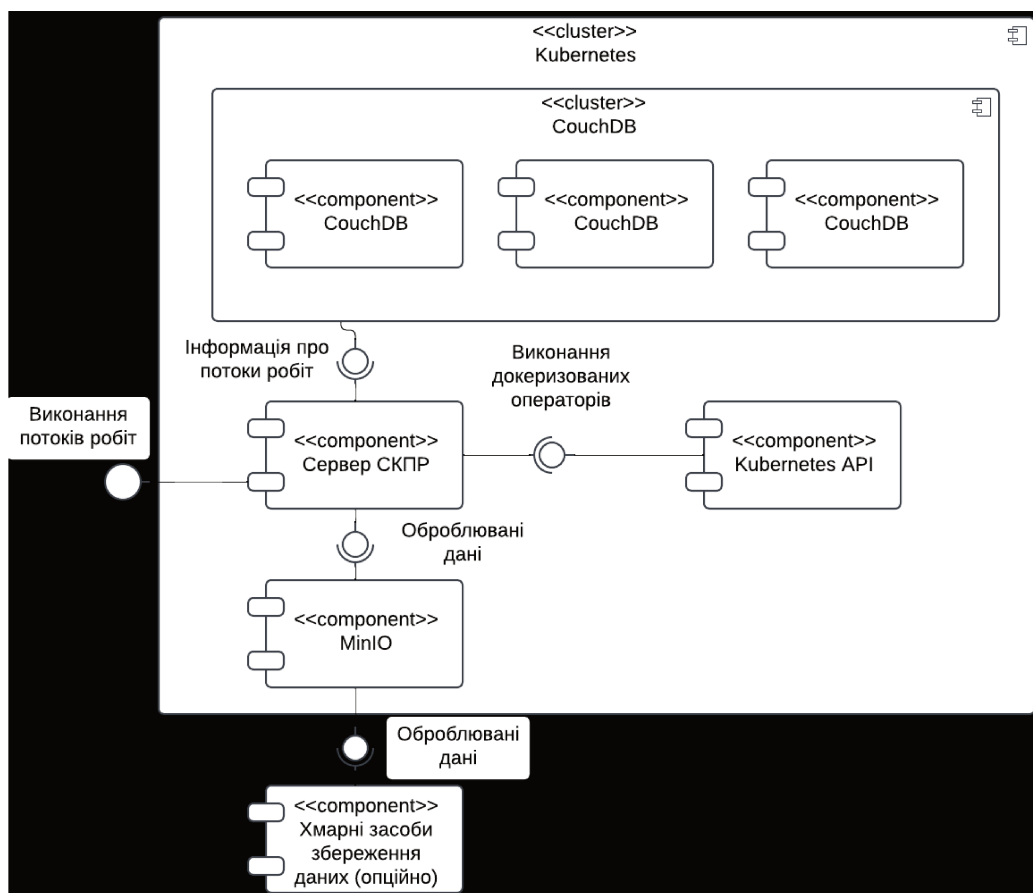


Рисунок 3.6 – Діаграма компонент СКПР

Розглянемо компоненти системи детальніше.

Систему керування потоками робіт реалізовано у вигляді HTTP-сервера на основі Node.js та фреймворку NestJS. Таке рішення було прийнято зважаючи на асинхронну будову Node.js, яка є важливою в даному контексті, оскільки більшість часу під час виконання ПР за допомогою даної СКПР система знаходиться або в стані очікування переміщення даних, або в стані очікування завершення обчислень. Вибір

фреймворку обґрунтовано абстрактністю та простотою використання, що фокусує реалізацію на розв'язанні задач реалізації алгоритму та організації даних та обчислень.

Збереження оброблюваних даних організовано за допомогою об'єктного сховища MinIO, яке надає інтерфейс до зберігання та видобування довільних файлів. Використання власного об'єктного сховища є ключовим для ізоляції СКПР від конкретної системи зберігання даних, що використовується експлуатантом. Також, MinIO надає можливість розподілення даних по рівнях (storage tiering) за часом використання – вивантаження даних в дешевші, але повільніші засоби зберігання, вивантажуючи невикористовувані дані для вивільнення швидшого та ближчого місця для нових даних або даних що часто використовуються. Серед альтернативних рішень MinIO було обрано через повну сумісність з стандартним для індустрії API Amazon S3 та відкриту ліцензію – GNU AGPL v3.

Збереження інформації про обчислення у системі організовано за допомогою документно-орієнтованої системи керування базами даних (СКБД). Дане рішення обґрунтовано пристосованістю таких СКБД до зберігання складних структур даних, їх швидкого запису та видобування. Відмова від використання традиційних реляційних баз даних не завдає втрат та ускладнень у випадку даної СКПР, оскільки всі наявні сутності мають сенс лише у зв'язку з головною сутністю засобу – потоком робіт. Таким чином, зберігання потоку робіт та усієї іншої інформації (кроки, одиниці даних тощо) що з ним пов'язана у вигляді одного документу надає переваги у простоті використання, швидкості запису, прочитання та реплікації даних між клонами бази. Враховуючи вимоги до горизонтального масштабування, було обрано документно-орієнтовану СКБД Apache CouchDB, спроектовану від початку для використання у кластерному вигляді – стандартне розгортання CouchDB містить три екземпляри бази даних, що синхронізуються між собою. Також, підтримка автоматичного горизонтального масштабування в CouchDB дозволяє бути впевненим в тому, що горизонтальне масштабування програмного засобу під навантаженням не буде обмежене відсутністю горизонтального масштабування бази даних.

Серед систем кластерних та ґрід-обчислень з підтримкою Docker можна виділити наступні:

- Kubernetes;
- HTCondor;
- Slurm.

Для практичної реалізації з наявних систем було обрано Kubernetes, оскільки він є де-факто стандартом для виконання кластерних обчислень та має широку нативну підтримку серед багатьох хмарних платформ.

Kubernetes API дозволяє залишати запити на виділення постійної пам'яті та планування і виконання завершуваних одноразових контейнеризованих обчислень. Таким чином, за допомогою використання Kubernetes API, а особливо Batch API, визначені алгоритмом кроки з визначеними операторами, їх вхідними даними та параметрами можуть бути обчислені шляхом запуску контейнеризованого коду у Kubernetes-задачах що завантажують вхідні дані з програмного засобу, виконують контейнеризоване обчислення оператора, та вивантажують результати обчислень до програмного засобу. Таким чином, СКПР на основі запропонованого методу ізолюється від розв'язання проблем, пов'язаних з плануванням та виділенням ресурсів, та фокусується виключно на керуванні даними та визначенні необхідних для проведення обчислень.

Відповідно, система керування потоками робіт розгортається на базі Kubernetes-кластеру, який містить довільну, залежну від навантаження, кількість екземплярів сервера програмного засобу, які підтримуються вбудованим в Kubernetes-кластер API-сервером, та горизонтально масштабовані систему об'єктного (файлового) сховища та документно-орієнтовану базу даних.

3.2.2 Програмна реалізація системи

СКПР реалізовано у вигляді HTTP-сервера на базі фреймворку NestJS для платформи Node.js. Відповідно, архітектура слідує типовій тришаровій архітектурі: «Контролер» => «Сервіс» => «Доступ до даних» (рис. 3.7).

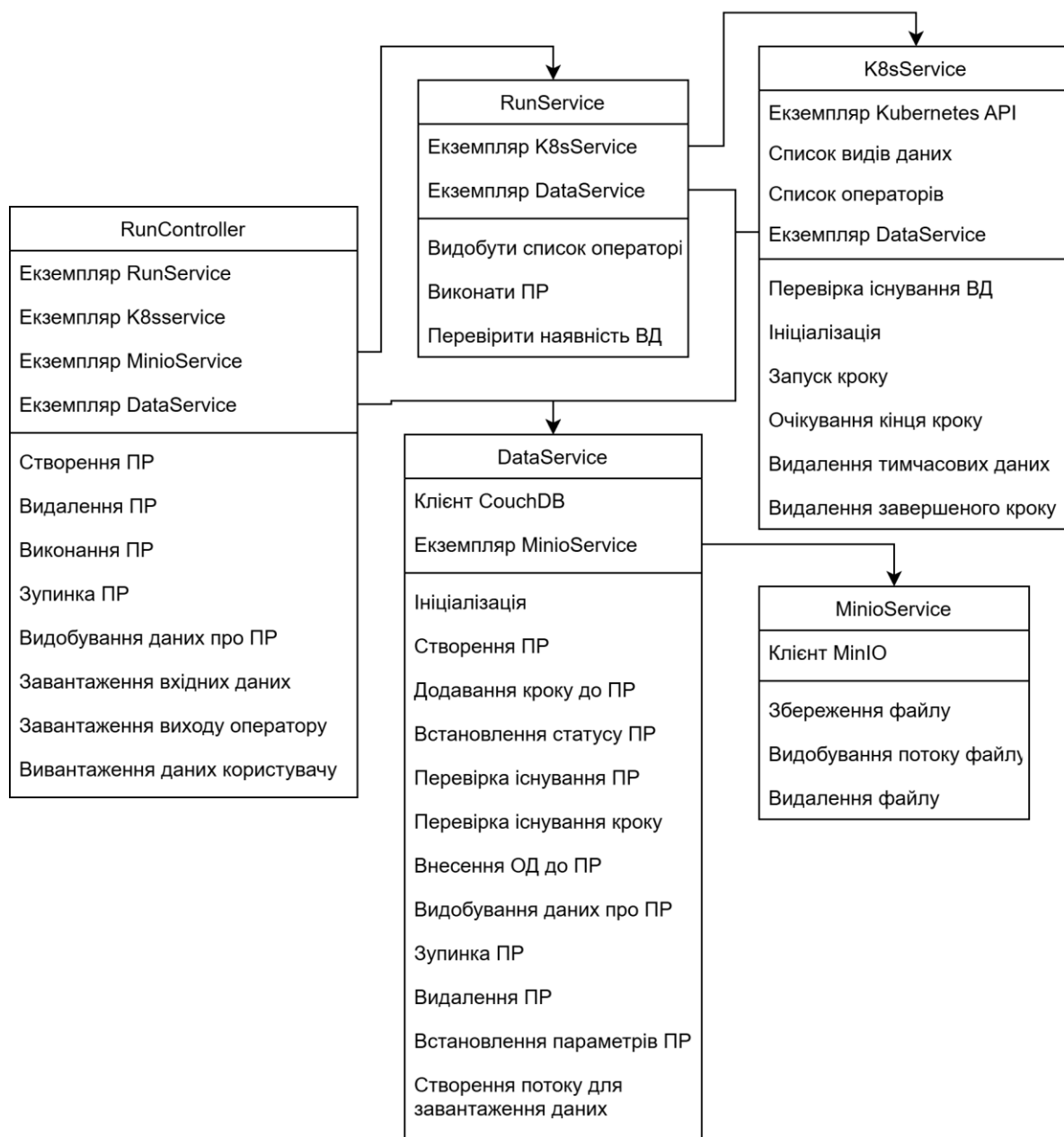


Рисунок 3.7 – Діаграма класів СКПП на основі запропонованого методу

Контролер відповідає за прийняття запитів, їх маршрутизацію до відповідного обробника. Обробники запиту представлені у вигляді методів Контролера, та видобувають необхідні для обробки запиту дані, організують бізнес-логіку шляхом виклику методів Сервісів та формують відповідь на запит за допомогою результатів роботи методів Сервісів. Сервіси реалізують бізнес-логіку, та визначають як дані створюються та змінюються. Шар доступу до даних (часто також виконується у вигляді сервісу) реалізує базові операції: створення, видобування, видозмінення,

видалення. Таким чином досягається ізоляція та інкапсуляція роботи з базою даних чи іншим сховищем інформації абстрактний інтерфейс.

Відповідно, СКПР складається з одного контролера (a), двох організаційних сервісів (b, c) та двох сервісів доступу до даних (d, e):

- a) RunController є основним контролером, що надає доступ до усіх операцій над ПР – створення, наповнення даними, запуск тощо;
- b) RunService містить реалізацію вищеописаного алгоритму, та повністю керує виконанням обчислень у ПР;
- c) KubernetesService реалізує операції, необхідні для планування задач та виконання обчислень в Kubernetes, та очікування на їх завершення;
- d) MinioService є частиною шару доступу до даних та надає доступ до об'єктного файлового сховища Minio для збереження даних, над якими здійснюються обчислення;
- e) DataService є частиною шару доступу до даних та надає доступ до зберігання, редагування та видалення інформації про потік робіт, елементів даних всередині ПР, кроків тощо.

Для забезпечення виконання кроків, знайдених алгоритмом, KubernetesService інкапсулює ряд викликів до Kubernetes API. Робота розпочинається з виклику методу runStep, який приймає інформацію, необхідну для створення кроку, створює необхідні Kubernetes-сутності та очікує завершення обчислень.

Відповідно, для уможливлення планування та виконання кроків за допомогою Kubernetes, Оператори є Docker-контейнерами, що містять необхідний для виконання обчислень код. Оператори приймають дані у вигляді файлів, розміщених на диску контейнера з оператором. Шляхи до файлів, а також запитувані параметри оператора задаються СКПР в змінних оточення контейнера оператора. Назви змінних вказуються при реєстрації оператора.

Для виконання запропонованого кроку з певним оператором, СКПР PersistentVolumeClaim – заброньоване місце на абстрагованому диску для збереження файлів в межах кластеру – та створює Kubernetes-задачу з трьох кроків – завантаження вхідних для Оператору даних з MinIO, проведення обчислень

Оператором, та вивантаження вихідних даних у MinIO крізь систему з прикріпленням цих даних до ПР, до якого належить крок та маркуванням даних видом даних, вказаним Оператором. Завантаження та вивантаження даних здійснюється окремим докеризованим інструментом. Таке рішення було прийнято оскільки в найбільших провайдерах хмарної інфраструктури, найширші права на використання PersistentVolumeClaim обмежені ReadWriteOnce/ReadOnlyMany – лише один под (англ. pod) (група контейнерів, які ділять між собою мережеві та сховищні ресурси) може мати право запису на віртуальний диск. Також, PersistentVolumeClaim можуть бути під'єднані до групи контейнерів лише при створенні поду, що виключає тимчасове монтування диску до контейнера, в якому запущено СКПР для запису даних на диск перед обчисленням та видобування даних після завершення обчислень.

Для роботи з документно-орієнтованою базою даних CouchDB було обрано використати клієнт CouchDB, однак без використання локалізованого клону бази даних, оскільки СКПР є орієнтованою на роботу всередині одного кластеру.

Усі дані про потік робіт зберігаються як поля об'єкта потоку робіт:

- прораховані кроки;
- статус їх виконання;
- наявні одиниці даних;
- місце зберігання ОД тощо.

Таким чином, робота з базою побудована навколо однієї CouchDB-таблиці, що значно спрощує роботу з базою, догляд за її станом, тощо. Важливо зауважити, що документи в CouchDB є версіонованими – і при кожній зміні документа необхідно надати версію, від якої ця зміна внесена, що допомагає помічати та попереджати паралельні зміни та конфлікти запису (дві сутності намагаються внести різні зміни водночас, відштовхуючись від однієї версії даних, що призводить до втрати узгодженості даних або втрати даних загалом). Оскільки в СКПР на основі запропонованого методу відбувається велика кількість паралельних обчислень (під час обчислення потоку робіт), результати і процес яких продукують зміни в один і той самий документ, імовірність появи конфліктів, та, відповідно, відмов бази у записі через невідповідність версії від якої створено зміну до останньої версії наявної в базі

є практично стовідсотковою. Для уникнення проблем такого роду, на базі сервісу який ізолює роботу з базою, було реалізовано примітивний транзакційний механізм, який перевидобуває оновлену версію документа з бази та повторює внесення зміни в цей документ, доки не буде досягнуто успішного збереження в базу без конфліктів.

3.2.3 Програмний інтерфейс взаємодії з СКПР

Для використання СКПР на основі запропонованого методу після її розгортання на Kubernetes-кластері необхідно:

- розробити оператори, які використовуватимуться для виконання обчислень;
- завантажити їх в Docker-реєстр (якщо реєстр приватний – необхідно також переконатись що Kubernetes-кластер є автентифікованим для завантаження з цього реєстру);
- оголосити їх в кластері разом з видами даних, які обробляються.

Оператори та види даних визначаються за допомогою визначення спеціальних Kubernetes-ресурсів (CRD – custom resource definitions), які стають доступними у Kubernetes-кластері після розгортання програмного засобу. Так, вид даних визначається як строкова мітка, яка додатково має строкову назву для використання у інтерфейсах:

```
apiVersion: mlynar.dev/v1alpha1
kind: DataKind
metadata:
  name: number-array-json
spec:
  displayName: 'Number Array JSON'
```

Для Оператора ж визначається посилання на Docker-зображення, аргументи що передаються до точки входу зображення, список видів даних, які можуть бути результатами обчислень оператора., а також змінні оточення:

Постійні змінні оточення – забезпечують можливість статичного конфігурування конкретного зображення. Так, наприклад, одне і те саме зображення з різними статичними змінними оточення може вимагати різних вхідних даних та відповідно використовуватись у різних операторах. Для таких змінних оточення задається назва змінної та її строкове значення;

Конфігуровані змінні оточення – слугують для можливості динамічного налаштування оператора, наприклад через інтерфейс користувача, за допомогою якого запускаються обчислення. Для таких змінних оточення задається назва змінної та значення за замовчуванням;

Вхідні дані – для кожного входу оператора визначається вид даних, що має бути наданий до цього входу, а також змінна оточення, в якій буде зазначено шлях до файлу, що представляє вхідні дані зазначеного виду.

Для прикладу, визначимо оператор, що обчислює середнє арифметичне для масиву чисел, послуговуючись довжиною масиву та сумою елементів масиву, що надійшли від інших операторів. У абстрактному записі даний оператор виглядає як `average(number-array-sum-json, number-array-length-json) => number-array-average-json`.

У вигляді Kubernetes-ресурсу опис оператора виглядає наступним чином:

```
apiVersion: mlynar.dev/v1alpha1
kind: Operator
metadata:
  name: average
  namespace: default
spec:
  image: "example/example-average:latest"
  possibleOutputKinds:
    - "number-array-average-json"
  inputs:
    - fileLocationEnv: "INPUT_SUM"
      dataKind: "number-array-sum-json"
```

```

- fileLocationEnv: "INPUT_LENGTH"
  dataKind: "number-array-length-json"
constantEnv:
- name: "EXAMPLE_ENV_VAR"
  defaultValue: "example"
configurableEnv:
- name: "MULTIPLY_BY"
  defaultValue: "1"

```

Для коректної роботи, точка входу зображення (entrypoint) мусить запускати загорнуте програмне забезпечення. Загорнуте програмне забезпечення видобуває свої статичну та динамічну конфігурації, а також розміщення файлів вхідних даних зі змінних оточення, оголошених під час реєстрації оператору у Kubernetes-ресурсі.

Після виконання обчислень, контейнеризований оператор зберігає результати на диску контейнера. Розміщення, в яке мають бути записані відповідні файли вказано в окремій змінній оточення – `MLYNAR_OUTPUT_DIR`. На додачу до вихідних файлів з результатами обчислень, програмне забезпечення оператору має записати файл-маніфест, який перелічує видані результати обчислень та маркує їх відповідними видами даних. Наприклад, оператор, який обчислює середнє арифметичне вищеописаного виду «Масив чисел», видає такий файл-маніфест:

```

{
  "outputs": [
    {
      "dataKind": "number-array-average-json",
      "file": "/data/output/average.json"
    }
  ]
}

```

Маніфест структуровано у вигляді об'єкта з одного поля задля забезпечення зворотної сумісності у випадку необхідності розширення кількості або виду полів-результатів роботи оператору. Таким чином, у випадку розширення структури цього

об'єкта для підтримки інших видів передачі результатів обчислення, або інших форм результату роботи оператору тощо, оператори, що використовують стару версію, все ще підтримуватимуться новішими версіями програмного забезпечення. Так само, об'єкт, що описує конкретний файл-результат дозволяє додавання уточнень і розширень в майбутньому зі збереженням зворотної сумісності. Також, варто доповнити, що результатом роботи оператору може бути довільна кількість файлів довільних видів даних (з зареєстрованих), всі з яких мають бути вказані в маніфесті результатів обчислення, однак для коректного передбачення операторів, що можуть бути запущені під час проведення обчислень над наданим набором одиниць даних, необхідно вказати всі можливі види даних, які можуть бути представлені оператором як результат обчислень.

Для виконання потоку робіт здійснюється наступна послідовність операцій:

- 1) створення ПР;
- 2) прикріплення даних;
- 3) параметризація ПР;
- 4) запуск ПР;
- 5) відстеження виконання ПР;
- 6) видобування результатів обчислень.

Дані операції виконуються за допомогою HTTP-запитів. Розглянемо дані запити детальніше. Для виконання обчислень зареєстрованими операторами необхідно створити потік робіт (зап. 1):

Запит 1: `POST /api/run`

Тіло: `пусте`

Відповідь: `{`

`"_id": "26d3d028-91c9-4f2f-8098-c2092a026a58",`

`"steps": [],`

`"status": "created",`

`"dataPool": [],`

`"paramPool": {}`

`}`

Ця операція створює необхідну сутність в базі даних. Після завершення цієї операції потрібно завантажити необхідні вхідні дані для обчислень зареєстрованими операторами, прикріпити їх до новоствореного ПР та охарактеризувати видом даних (зап. 2):

```
Запит 2: POST /api/run/{id}/upload/{dataKind}
Тіло: FormData, що містить файл у довільному полі
Відповідь: {
  "id": "d3b138aa-9661-4a08-a9db-831b7bf36e08",
  "dataKind": "{dataKind}",
  "ancestors": []
}
```

Оскільки оголошені оператори мають параметри, їх можна видобути скориставшись наступним запитом (зап. 3). Отримані дані про параметри можуть бути використані так і для конфігурації кінцевим користувачем у графічному інтерфейсі, так і для автоматичної конфігурації іншим програмним забезпеченням.

```
Запит 3: GET /api/run/{id}/params
Відповідь: {
  "operatorA": {
    "paramAA": "defaultValueAA",
    "paramAB": "defaultValueAB",
  },
  "operatorB": {
    "paramBA": "defaultValueBA",
    "paramBB": "defaultValueBB",
  },
}
```

Після вибору значень параметрів операторів, вони можуть бути передані у наступний запит (зап. 4) для ініціювання проведення обчислень. При виконанні цього

запиту передані параметри операторів буде записано до обраного потоку робіт, та буде розпочато його виконання. Статус ПР зміниться з “created” на “running”.

Запит 4: POST /api/run/{id}/run

Тіло: відповідь попереднього запиту з обраними користувачем значеннями

```
Відповідь: {
  "_id": "26d3d028-91c9-4f2f-8098-c2092a026a58",
  "steps": [],
  "status": "running",
  "dataPool": [{
    "id": "d3b138aa-9661-4a08-a9db-831b7bf36e08",
    "dataKind": "{dataKind}",
    "ancestors": []
  },
  ...],
  "paramPool": {тіло запиту}
}
```

Для перевірки статусу виконання ПР, а також списку кроків, їх статусу – «створено», «обчислюється», «успішно обчислений», «обчислений з помилкою», списку наявних у ПР Одиниць даних – як вхідних, так і доданих в процесі обчислення, застосовується наступний запит. Передбачається, що система, що використовуватиме СКПР перевірятиме статус обчислення ПР шляхом виконання окремих запитів, внаслідок дії користувача або за часовим інтервалом.

Запит 5: GET /api/run/{id}/run

```
Відповідь: {
  "_id": "26d3d028-91c9-4f2f-8098-c2092a026a58",
  "steps": [{
    "id": "c8387a43-b1f8-4128-a0c1-26488621ff5e",
    "inputDataUnits": {
      "input1": "d3b138aa-9661-4a08-a9db-831b7bf36e08"
    }
  }
]
```

```

    },
    "outputDataUnits": [
        "49e7261c-771e-4f12-a456-e95f1da9c884"
    ],
    "operator": "operatorA",
    "status": "running"
...],
"status": "running",
"dataPool": [{
    "id": " d3b138aa-9661-4a08-a9db-831b7bf36e08",
    "dataKind": "{dataKind}",
    "ancestors": []
}],
"paramPool": {тіло запиту}
}

```

Для передчасної зупинки ПР у разі зникнення потреби у проведенні обчислень, рішенні зміни параметрів чи вхідних даних або досягнення необхідного результату обчислень, може бути виконаний наступний запит (зап. 6):

Запит 6: POST /api/run/{id}/terminate

Тіло: пусте

Відповідь: {

```

    "message": "Run terminated",

```

```

}

```

До, під час та після завершення виконання ПР, будь-яка одиниця даних може бути видобута з системи за ідентифікатором (зап. 7):

Запит 7: GET /api/run/data-unit/{dataUnitId}

Відповідь: Файл одиниці даних

Для повного очищення системи від даних (як і даних про сам ПР, так і вхідних, проміжних та вихідних даних обчислень ПР), пов'язаних з певним ПР, використовується наступний запит (зап. 8):

```
Запит 8: DELETE /api/run/{id}
```

```
Відповідь: {
  "message": "Run deleted",
}
```

Таким чином, запропонована СКПР пропонує повний набір операцій для запуску потоків робіт, відстежування їх виконання, видобування результатів обчислених операторів та видалення даних.

3.3 Аналіз швидкодії СКПР

Оцінка швидкодії системи впливає на загальну оцінку самого підходу.

Розглянемо часові характеристики використання СКПР на основі запропонованого методу для проведення обчислень. Його можна розділити на наступні компоненти:

- час, що витрачається на корисні обчислення;
- накладні витрати на роботу алгоритму виявлення наступних кроків;
- накладні витрати на підготовку оточення виконання кроку та передачу вихідних даних;

Основним компонентом загального часу виконання потоку робіт є час, що витрачається на проведення корисних обчислень (3.1):

$$t_{\text{кор.}} = \sum_{i=0}^n t_{\text{кроку } i}, \quad (3.1)$$

де:

- $t_{\text{кор.}}$ є загальним часом корисних обчислень;

- n є кількістю окремих кроків ПР;
- $t_{\text{кроку}}$ є часом обчислення окремого кроку ПР.

Час, що витрачається алгоритмом СКПР на виявлення наступних кроків можна представити наступним чином (3.2):

$$t_{\text{алг.}} = \sum_{i=0}^n t_{\text{ітерації } i}, \quad (3.2)$$

де:

- $t_{\text{алг.}}$ є загальним часом роботи алгоритму;
- n є кількістю окремих ітерацій алгоритму;
- $t_{\text{ітерації}}$ є часом обчислення окремої ітерації алгоритму.

Аналогічно виглядають і накладні витрати на підготовку оточення кроків і передачу даних (3.3):

$$t_{\text{під.}} = \sum_{i=0}^n t_{\text{оточення } i}, \quad (3.3)$$

де:

- $t_{\text{зап.}}$ є загальним часом на роботу з оточенням;
- n є кількістю окремих кроків ПР;
- $t_{\text{оточення}}$ є часом на підготовку оточення окремого кроку ПР.

Таким чином, загальний час виконання потоку робіт є сумою часу корисних обчислень, часу роботи алгоритму та часу на роботу з оточенням:

$$t_{\text{заг.}} = t_{\text{кор.}} + t_{\text{алг.}} + t_{\text{під.}}, \quad (3.4)$$

Час $t_{\text{алг.}}$ та $t_{\text{під.}}$ залежить від способу реалізації СКПР (оцінюється в ізоляції або у складі комплексної системи).

3.3.1 Аналіз швидкодії алгоритму в ізоляції

За умови, що накладні витрати часу, що необхідні для динамічного конструювання алгоритмом графу потоку робіт під час його виконання є істотно меншими за час, що витрачається на корисні обчислення та роботу з оточенням (3.4):

$$t_{\text{кор.}} + t_{\text{під.}} \gg t_{\text{алг.}} \quad (3.5)$$

часом роботи алгоритму можна знехтувати (3.5):

$$t_{\text{заг.}} \approx t_{\text{кор.}} + t_{\text{під.}} \quad (3.6)$$

При виконанні даної умови також зберігатимуться переваги алгоритму у економії часу на розробку та модифікацію ПЗ що використовує потоки робіт шляхом зменшення кількості втручань у код, необхідних для модифікації виконуваного у потоці робіт коду.

Перевіримо виконання (3.5) шляхом заміру часу його роботи на потоках робіт різного розміру. Оскільки суть алгоритму полягає у динамічному створенні потоків робіт під час їх виконання, тестування було виконано шляхом рандомізованого створення наборів з операторів, видів даних та вхідних одиниць даних.

Окрім замірів часу у потоках робіт різної складності, тестування таким шляхом дозволяє пересвідчитися у здатності алгоритму витримувати застосування у довільному оточенні для вирішення довільних обчислювальних задач.

Для створення реалістичних штучних оточень та потоків робіт було реалізовано набір функцій, що виконують повну підготовку тестових даних. Для кожного тестувального ПР генерується 10 випадкових видів даних та 15 випадкових операторів. Для кожного оператора випадковим чином обирається один чи два входи, які можуть бути обрані:

- з першої третини видів даних;
- з другої третини видів даних;

з імовірністю вибору як між кількістю входів, так і між третинами у 50%. Вибір виду даних всередині третини є рівномірним. Один (з імовірністю 80%) чи два (з імовірністю у 20%) вихідних види даних оператору обираються з останніх двох третин видів даних. Такий підхід було обрано для імітації більш реалістичних сценаріїв використання з декількома зв'язаними потоками робіт, що мають вхідні, проміжні та вихідні дані.

Для ініціації потоку робіт створюється 3 одиниці даних довільних видів з згенерованих 10, після чого наступні кроки ПР обчислюються алгоритмом, ідентично до проведення реальних обчислень.

Оскільки виконання відбувається в штучному середовищі, після отримання наступних кроків в кінці кожного циклу алгоритму, обирається 95% обчислених кроків для утворення штучних «результатів» їх обчислень. Для кожного з цих кроків, кожен з їх можливих видів даних може бути повернутий з імовірністю у 60%, задля імітації прийняття логічних рішень всередині оператору. Для кожного повернутого виду даних створюється одна одиниця даних, яка повертається в набір даних ПР.

Під час виконання експериментів за описаною схемою істотна частина ПР, через випадкове генерування наборів операторів та видів даних, мала тенденцію до експоненційного розширення – збільшення кількості нових елементів даних, нових операторів, чим суттєво ускладнює проведення планувальних обчислень. Відтак, такі потоки робіт були відсіянні – якщо кількість ітерацій алгоритму перебільшує 30 чи кількість елементів даних перебільшує 1000 – ПР відкидається.

Попри те що основною причиною такої поведінки є випадкове та штучне створення операторів та їх взаємозв'язків, ненавмисне створення експоненційно зростаючих ПР у ситуаціях реального застосування є теоретично можливим.

В першу чергу, це вказує на необхідність в уважному ставленні до загального набору операторів, що здійснюють обчислення в системі, їх вхідних та вихідних даних, та у продумуванні можливих ситуацій експоненційного зросту.

В другу чергу, це вказує на простір для потенційного покращення підходу – введення попереднього статичного аналізу вхідних даних та наявних операторів на ризик раптового експоненційного зросту.

Було обчислено 523 штучні потоки робіт та зібрано дані про них (рис. 3.8).

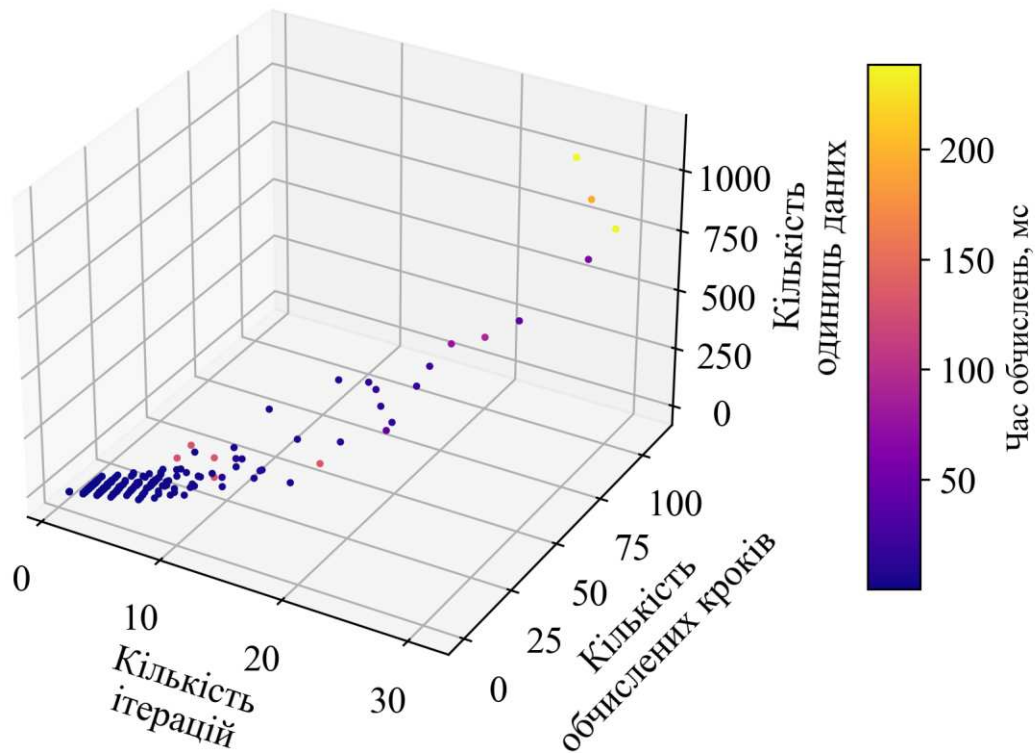


Рисунок 3.8 – Потоки робіт, отримані в результаті обчислювального експерименту

Отримані в результаті експерименту потоки робіт мали істотно різні параметри:

- кількість ітерацій алгоритму від 1 до 31;
- кількість кроків від 0 до 110;
- одиниць даних від 0 до 1143;
- час, що витрачено на обчислення від <1мс до 238.47мс.

Відтак, аби аналіз над отриманими вимірами був змістовним, отримані потоки робіт було розділено на 4 кластери за допомогою алгоритму k-найближчих сусідів. Кількість кластерів було обрано емпірично, базуючись на візуальному розподілі даних та можливості зіставлення отриманих груп з реальними сценаріями використання ПР. Таким чином, було отримано групи що представляють потоки робіт суттєво різних розмірів.

Порівняння між отриманими кластерами по кроках, кількості одиниць та кількості ітерацій алгоритму наведено на рис. 3.9.

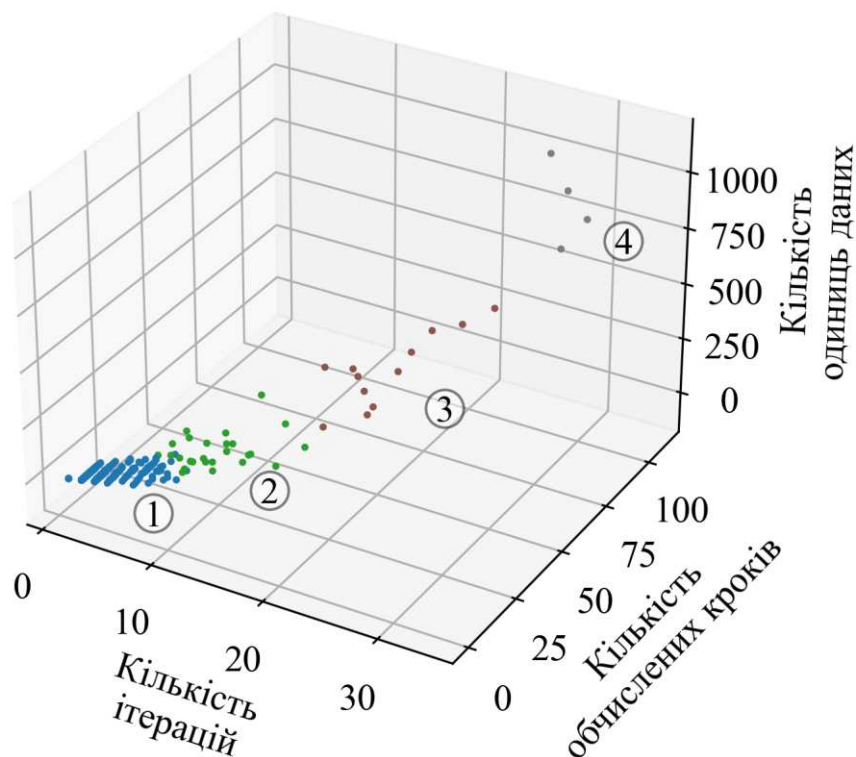


Рисунок 3.9 – Поточки робіт, отримані в результаті обчислювального експерименту, розділені на 4 кластери (k-найближчих)

Узагальнену статистичну інформацію про виявлені групи симульованих у випадково згенерованому середовищі ПР наведено у табл. 3.1.

Таблиця 3.1 – Дані про поведінку алгоритму у тестовому середовищі

| | Кластер 1 | Кластер 2 | Кластер 3 | Кластер 4 |
|---------------------------|-----------|-----------|-----------|-----------|
| Кількість потоків робіт | 479 | 28 | 12 | 4 |
| Ітерацій алгоритму (мін) | 1 | 6 | 16 | 27 |
| Ітерацій алгоритму (макс) | 9 | 17 | 27 | 31 |
| Ітерацій алгоритму (сер) | 2.77 | 10.14 | 20.83 | 29.00 |
| Ітерацій алгоритму (мед) | 2 | 10 | 21 | 29 |
| Обчислених кроків (мін) | 0 | 13 | 28 | 87 |
| Обчислених кроків (макс) | 20 | 47 | 70 | 110 |
| Обчислених кроків (сер) | 5.09 | 22.57 | 51.00 | 96.25 |
| Обчислених кроків (мед) | 4 | 21 | 49 | 94 |
| Одиниць даних (мін) | 0 | 43 | 283 | 793 |

| | | | | |
|--------------------------|--------|--------|--------|--------|
| Одиниць даних (макс) | 42 | 153 | 612 | 1143 |
| Одиниць даних (сер) | 5.83 | 82.21 | 391.08 | 916.00 |
| Одиниць даних (мед) | 3 | 75 | 369 | 864 |
| Час обчислень, мс (мін) | 0.0089 | 1.78 | 10.95 | 61.86 |
| Час обчислень, мс (макс) | 6.34 | 133.89 | 91.55 | 238.47 |
| Час обчислень, мс (сер) | 0.58 | 27.56 | 30.82 | 182.25 |
| Час обчислень, мс (мед) | 0.37 | 4.55 | 16.44 | 214.33 |

За даними, наведеними у табл. 3.1 та рис. 3.8-3.10 видно, що за результатом кластеризації за допомогою алгоритму k-найближчих було виділено 4 кластери потоків робіт різної обчислювальної складності, які відповідають реалістичним задачам, що можуть бути поставлені перед алгоритмом:

- кластер 1, найбільший з представлених, можемо вважати відповідником найбільш звичайним та поширеним потокам робіт, таким як біоінформатичні дослідження [98] [99] та інші – до 20 окремих обчислень, та до 7мс витрати на планування кроків, з 5 кроками та до 1мс часу в середньому;
- кластер 2 – представляє середньо-великі обчислювальні навантаження такі, як, наприклад, обробка астрономічних даних [100] чи детальний аналіз секвенованого РНК (RNA-Seq) [101], до 47 кроків та 134мс та 22 кроки і 28мс у середньому;
- кластери 3 та 4, які відповідають великим та надвеликим потокам робіт з усіх сфер науки, включаючи біоінформатичні дослідження з реконструкції транскриптом без референсного геному [102], до (у симуляції, на практиці необмежено) 110 кроків та 238мс.

Обчислення диференційної експресії генів (що є типовим раннім кроком в геномних дослідженнях) може займати від однієї хвилини (часто декількох хвилин) процесорного часу навіть на високопродуктивних обчислювальних системах [103]. Обчислення вкладених представлень знімків КТ на наявному обладнанні займає біля 300мс для кожного зрізу.

Відтак, наведені у таблиці дані про об'єм та швидкість обчислень показують що у кожному з виявлених кластерів умова (3.5) виконується.

Обчислювальні експерименти проводились на екземплярі програмного засобу, розгорнутому на Kubernetes-кластері Google Cloud, що складається з однієї віртуальної машини типу shared-core (одне ядро процесора ділиться між декількома віртуальними ядрами) e2-medium, з 940 mCPU та 3 ГБ оперативної пам'яті максимального запиту.

Експерименти проводились на обмеженій в обчислювальній потужності віртуальній машині при наявності у кластері інших контейнерів які є частиною програмного засобу (база даних, об'єктне сховище). Крім того, обчислювальні потужності обслуговували також контролюючий шар самого Kubernetes-кластеру. Відтак, обчислення для вирішення задачі конкурували за ресурси з іншими процесами. При використанні кластера, що забезпечує розподілення керуючих функцій, ефективний час використання обчислювальних потужностей на вирішення корисної задачі буде зростати, а відтак при однакових технічних параметрах обчислювальної системи час, що необхідний алгоритму для обчислення наступних кроків буде меншим.

3.3.2 Аналіз швидкодії СКПР у комплексі

Швидкодію СКПР у комплексі було протестовано шляхом запуску потоків робіт з плануванням реальних обчислень, послідовно та паралельно. Таке тестування дозволяє заміряти накладні витрати часу, такі як:

- виділення ресурсів кластера;
- підготовка даних для обчислень;
- запуск контейнерів;
- завантаження даних назад у сервер;
- вивільнення ресурсів.

Для проведення тестування було обрано набір простих видів даних:

- масив чисел;
- довжина масиву чисел;
- сума масиву чисел;
- середнє арифметичне масиву чисел;

та операторів:

- обчислювач довжини масиву (масив => довжина);
- обчислювач суми масиву (масив => сума);
- обчислювач середнього арифметичного (довжина, сума => середнє).

Сформований з даних ВД та операторів потік робіт виглядає наступним чином (рис. 3.10).

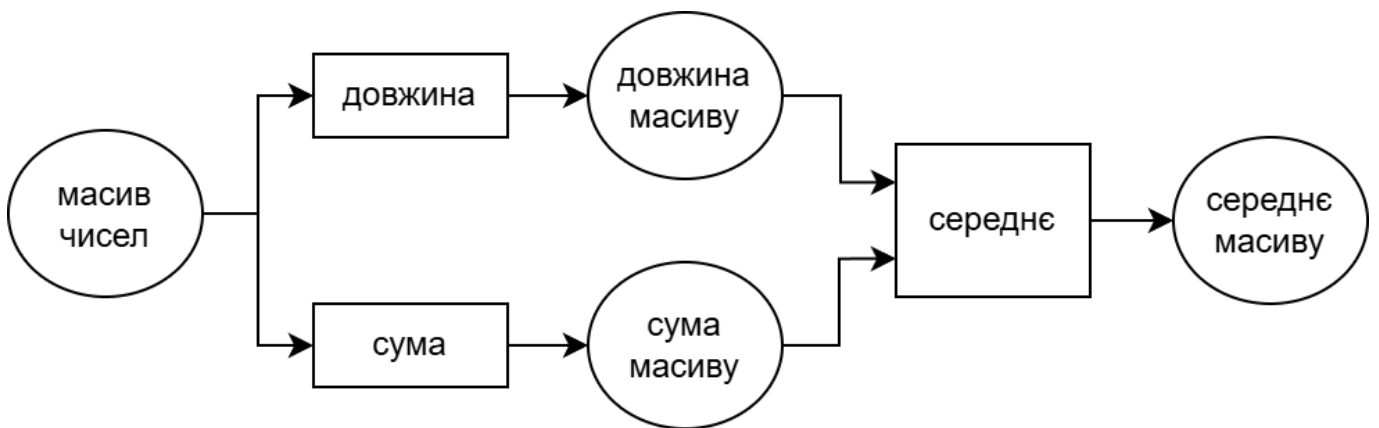


Рисунок 3.10 – Потік робіт обчислення середнього арифметичного масиву чисел

Операції, виконувані у цьому ПР є елементарними та швидкими. Кожна з цих операцій ізольована в окремому операторі, і, відтак, вимагає окремої підготовки даних та завантаження їх до сервера. Вхідний масив чисел було обрано коротким для зменшення участі часу обчислень у загальному часі виконання ПР. Відтак, з обраними вхідними даними, усі виконувані операції сумарно займають менше однієї мілісекунди, тому можемо вважати що час, отриманий під час тестування є накладним часом.

Накладний час було виміряно у двох сценаріях:

1. послідовне виконання ПР у кластері в один вузол;
2. паралельне виконання ПР у кластері в п'ять вузлів.

Такі сценарії було обрано для фіксування швидкодії системи як у проведенні індивідуальних обчислень, так і у проведенні декількох обчислень водночас, і, таким чином, фіксування різниці у швидкості алокації ресурсів при появі декількох одночасних запитів на ці ресурси.

Для проведення вимірів було розроблено скрипт, що створює ПР, завантажує до нього дані, ініціює ПР та з інтервалом у 500 мілісекунд опитує сервер до завершення виконання потоку робіт. Для сценарію 1, після завершення одного потоку робіт, відразу запускається наступний до досягнення потрібної кількості експериментів. Для сценарію 2 декілька ПР запускається водночас, та по завершенню будь-якого з них запускається наступний для підтримки постійної кількості виконуваних обчислень, до досягнення кінцевої кількості завершених ПР.

Для першого сценарію було послідовним чином обчислено 100 потоків робіт, та заміряно час від запуску кроку до його закриття (включно з деалокацією ресурсів).

Інформацію про отримані виміри наведено у таблиці 3.2.

Таблиця 3.2 – Виміри швидкості обчислень, послідовно, у кластері в 1 вузол

| | Потік робіт загалом | Кроки окремо |
|---------------------------|---------------------|--------------|
| Час обчислення (мін) (с) | 32.10 | 15.11 |
| Час обчислення (макс) (с) | 57.00 | 35.32 |
| Час обчислення (сер) (с) | 40.99 | 19.58 |
| Час обчислення (мед) (с) | 42.09 | 20.17 |

Для другого сценарію було обчислено 200 потоків робіт, з паралелізмом у 10 в будь-який момент часу обчислень. Аналогічно, для кожного кроку було виміряно час від запуску кроку до його завершення. Однак, на відміну від попереднього варіанту, кластер було розширено до 5 вузлів, розгорнутих на віртуальних машинах типу e2-medium.

Результати вимірів паралелізованих обчислень вказано у таблиці 3.3.

Таблиця 3.3 – Виміри швидкості обчислень, паралельно по 10, у кластері в 5 вузлів

| | Потік робіт загалом (с) | Кроки окремо (с) |
|-----------------------|-------------------------|------------------|
| Час обчислення (мін) | 32.80 | 15.09 |
| Час обчислення (макс) | 63.20 | 36.17 |
| Час обчислення (сер) | 47.51 | 21.87 |
| Час обчислення (мед) | 47.90 | 20.57 |

Обчислювальні експерименти проводились на екземплярі програмного засобу, розгорнутому на Kubernetes-кластері Google Cloud, що складається з однієї віртуальної машини типу shared-core (одне ядро процесора ділиться між декількома віртуальними ядрами) e2-medium, з 940 mCPU та 3 ГБ оперативної пам'яті максимального запиту.

Розглянемо співвідношення між накладними витратами часу на організацію оточення виконання оператора та передачу даних і загальним часом виконання потоку робіт (3.7):

$$\alpha = \frac{t_{\text{під.}}}{t_{\text{заг.}}} = \frac{t_{\text{під.}}}{t_{\text{під.}} + t_{\text{кор.}}} \quad (3.7)$$

Обчислимо це співвідношення на прикладі виконання кроку, що містить оператор який обчислює вкладені представлення зрізів знімка КТ. Для даної системи було визначено, що на обчислення ВП одного зрізу витрачається біля 0.3с. Знімки КТ з використаних даних мають біля 80-100 зрізів, що містять легені. Таким чином, час на побудову ВП усіх зрізів знімка коливається між $80 * 0.3 = 24с$ та $100 * 0.3 = 30с$. Таким чином, середній час на обчислення ВП усіх зрізів знімка визначається як середнє між максимальним та мінімальним $\frac{24+30}{2} = 27с$. За час на запуск кроку візьмемо середнє значення у 22 секунди з табл. 3.3.

Таким чином, співвідношення між корисним обчисленням та обчисленням оператору, що генерує ВП становить (3.8):

$$\frac{t_{\text{під.}}}{t_{\text{зар.}}} = \frac{22}{22 + 27} = \frac{22}{49} \approx 0.44 \quad (3.8)$$

Однак, для задач з більш тривалими обчисленнями дана частка зменшуватиметься, що у екстремальних випадках робитиме час на підготовку оточення незначущим (3.9):

$$\lim_{t_{\text{кор.}} \rightarrow \infty} \frac{t_{\text{під.}}}{t_{\text{зар.}}} = \lim_{t_{\text{кор.}} \rightarrow \infty} \frac{t_{\text{під.}}}{t_{\text{під.}} + t_{\text{кор.}}} = \lim_{t_{\text{кор.}} \rightarrow \infty} \frac{\frac{t_{\text{під.}}}{t_{\text{кор.}}}}{\frac{t_{\text{під.}}}{t_{\text{кор.}}} + 1} = \lim_{t_{\text{кор.}} \rightarrow \infty} \frac{0}{0 + 1} = 0 \quad (3.9)$$

Недоліком даної системи, за результатами вимірів, є істотні витрати часу на алокацію ресурсів та підготовку даних. Основною причиною величини даних накладних витрат є те, що СКПР ізолює оператори від складності роботи з рештою системи шляхом підготовки вхідних файлів на диску та вивантаження вихідних файлів з диску, що спрощує реалізацію операторів. Дані дії виконуються двома docker-контейнерами, час на виділення ресурсів та запуск яких і створює значну частину накладних розтрат часу.

Відтак, не рекомендується використання даної СКПР в поточному стані для обчислень, що є принаймні на порядок швидшими за час, необхідний для підготовки оточення обчислень. Однак, для обчислень що є співставними або більшими за часом виконання (3.7) (3.8), гнучкість підходу, використаного у даному програмному засобі є істотним аргументом на перевагу використання даної СКПР.

3.4 Аналіз часових характеристик використання систем керування потоками робіт з динамічною побудовою графу під час виконання

Для оцінки ефективності використання реалізованої СКПР з точки зору розробки та впровадження програмних засобів на її основі необхідно порівняти витрати часу на розробку статичного потоку робіт з визначенням операторів для динамічної побудови графу.

Розглянемо часові компоненти процесу побудови потоку робіт:

- $t_{\text{інт.}}$ є часом на визначення інтерфейсу окремих операторів;
- $t_{\text{роз.}}$ є часом на розробку окремих операторів;
- $t_{\text{комп.}}$ є часом на компонування потоку робіт.

Відтак, час на побудову статичного потоку робіт виглядає наступним чином (3.10):

$$t_{\text{стат.роз.}} = t_{\text{інт.}} + t_{\text{роз.}} + t_{\text{комп.}}, \quad (3.10)$$

– де $t_{\text{стат.роз.}}$ є повним часом на розробку статичного потоку робіт.

У порівнянні, при використанні динамічної побудови графу під час виконання ПР, елемент $t_{\text{комп.}}$ зникає (3.11):

$$t_{\text{дин.роз.}} = t_{\text{інт.}} + t_{\text{роз.}}, \quad (3.11)$$

– де $t_{\text{дин.роз.}}$ є повним часом на розробку динамічного потоку робіт.

При виникненні потреби у новому функціоналі при використанні звичайної СКПР, необхідно повне оголошення додаткового графу (навіть при перевикористанні окремих операторів, окрім перевикористання підграфів) (3.10). Однак, для системи на основі запропонованого методу необхідним є лише додавання нових операторів (3.11). Попередні оператори будуть перевикористані автоматично.

Розглянемо часові компоненти модифікації інтерфейсу існуючого(-их) оператору(-ів) (наприклад, реалізації функціоналу, що породжує додаткові входи чи виходи оператору), що використовується у декількох потоках робіт:

- $t_{\text{мод.оп.}}$ є часом на модифікацію реалізації оператору(-ів);
- $t_{\text{мод.інт.}}$ є часом на модифікацію інтерфейсу оператору(-ів);
- $t_{\text{узг.}}$ є часом на узгодження потоків робіт, у яких міститься оператор(-и) з проведеною модифікацією (3.12).

$$t_{\text{узг.}} = \sum_{i=1}^n t_{\text{узг. } i} \quad (3.12)$$

де:

- $t_{\text{узг. } i}$ є часом на узгодження потоку робіт з проведеною модифікацією;
- n є кількістю потоків робіт що використовують даний(-і) оператор(-и).

Відтак, процес модифікації оператору(-ів) статичного потоку робіт виглядає наступним чином:

$$t_{\text{стат.мод.}} = t_{\text{мод.інт.}} + t_{\text{мод.оп.}} + t_{\text{узг.}} \quad (3.13)$$

– де $t_{\text{стат.мод.}}$ є повним часом на проведення модифікації оператору (-ів) статичного потоку робіт.

Однак, при використанні СКПР на основі запропонованого методу, узгодження потоків робіт не є необхідним, оскільки ті будуються автоматично (3.13):

$$t_{\text{дин.мод.}} = t_{\text{мод.інт.}} + t_{\text{мод.оп.}} \quad (3.14)$$

– де $t_{\text{дин.мод.}}$ є повним часом на проведення модифікації оператора (-ів) динамічного потоку робіт.

Розглянемо різницю між часовими витратами на розробку та модифікацію між типовою та розробленою системами (3.15-3.16):

$$\begin{aligned} \Delta t_{\text{роз}} &= t_{\text{стат.роз.}} - t_{\text{дин.роз.}} = \\ &= (t_{\text{інт.}} + t_{\text{роз.}} + t_{\text{комп.}}) - (t_{\text{роз.}} + t_{\text{інт.}}) = \\ &= t_{\text{комп.}} \geq 0 \end{aligned} \quad (3.15)$$

$$\begin{aligned} \Delta t_{\text{мод}} &= t_{\text{стат.мод.}} - t_{\text{дин.мод.}} = \\ &= (t_{\text{мод.оп.}} + t_{\text{мод.інт.}} + t_{\text{узг.}}) - (t_{\text{мод.оп.}} + t_{\text{мод.інт.}}) = \\ &= t_{\text{узг.}} \geq 0 \end{aligned} \quad (3.16)$$

Таким чином, можемо стверджувати, що (3.17-3.18):

$$t_{\text{стат.роз.}} > t_{\text{дин.роз.}} \quad (3.17)$$

$$t_{\text{стат.мод.}} > t_{\text{дин.мод.}} \quad (3.18)$$

що доводить доцільність використання запропонованого методу динамічної побудови графів потоків робіт під час їх виконання та алгоритму, що його реалізує. Результати також підтверджують ефективність СКПР на основі запропонованого методу. Дані рішення можуть бути застосовані у системах, які містять ресурсоемні обчислення та мають потребу у пришвидшенні та спрощенні процесу впровадження змін.

3.5 Висновки до розділу

У даному розділі розроблено метод для динамічної побудови графу потоку робіт під час його виконання, який дозволяє уникнути необхідності у статичному визначенні потоку робіт та зменшує час на адаптацію програмного забезпечення при впровадженні змін.

Для запропонованого методу визначено основні сутності (види даних, оператори, одиниці даних, кроки), що забезпечують покриття усіх задач пов'язаних з формуванням потоку робіт. На кожній ітерації виконується пошук комбінацій «оператор-дані», що ще не були виконані, з урахуванням як вхідних у даних, так і тих, що є результатами обчислень. На відміну від існуючих, даний метод дозволяє зменшити за рахунок надлишковості обчислень час, що необхідний розробнику на модифікацію виконуваного коду.

Розроблено математичну модель обрахунку часу виконання. При імітаційному моделюванні показано, що час на виконання надлишкових обчислень динамічної побудови графу значно менший за час на виконання корисних обчислень. Тому, при практичному використанні, часом, що витрачається на надлишкові обчислення можна знехтувати.

Розроблено математичну модель оцінки часу на побудову потоків робіт за запропонованим методом та наведено умови, що визначають межі ефективності його використання.

Недоліком СКПР, заснованої на запропонованому методі, можна вважати додаткові накладні витрати часу на підготовку оточення виконання для оператора, оскільки система комунікує з операторами виключно за допомогою файлів. Такий підхід дозволяє спростити реалізацію, але сповільнює час обміну даними.

Розроблено систему керування потоками робіт, що використовує запропонований метод, з використанням системи Kubernetes для розподілених кластерних обчислень та можливістю горизонтального масштабування. Запропонований метод та система керування потоками робіт спрямовані на прискорення адаптації існуючих систем аналізу зображень для діагностики.

Розроблена система рекомендується до використання для, наприклад, біоінформатичних обчислень у дослідницьких організаціях, для яких властиві постійні зміни та модифікації кодової бази при проведенні досліджень.

4 ПРОГРАМНИЙ ЗАСІБ АНАЛІЗУ ЗНІМКІВ КТ ДЛЯ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ

Для практичного використання методів аналізу знімків КТ для діагностики захворювань було розроблено програмний засіб з графічним інтерфейсом користувача, який дозволяє застосувати розроблені методи до наданих користувачем знімків.

Програмний засіб, що реалізує запропоновані підходи та методи, забезпечує наступний функціонал:

- авторизація та аутентифікація з закритою реєстрацією;
- розділення доступу до інформації;
- завантаження знімка КТ;
- проведення аналізу знімка КТ;
- отримання результатів аналізу;
- пошук схожих випадків за близькістю векторів вкладених представлень.

Також, програмний засіб задовольняє наступні вимоги:

- мультитенантність – можливість створення організацій, всередині яких користувачі мають спільний доступ до даних та проведених аналізів, однак не мають доступу до даних та аналізів інших організацій;
- обмежений доступ – доступ до системи користувачі виключно за попередньою реєстрацією адміністратором ПЗ;
- портативність – мінімізація залежності від конкретних хмарних провайдерів, та має бути готовим до розгортання на будь-якому з них, а також на власних апаратних потужностях;
- масштабованість – здатність до залучення додаткових ресурсів при зміні навантаження на систему;
- надійність – коректна обробка помилок виконуваних обчислень.

Для програмного засобу визначено:

- загальний порядок взаємодії з ПЗ;

- головні залежності та архітектуру розгортання;
- оператори СКПР на основі запропонованих методів класифікації зображень КТ;
- сервер програмного засобу, що надає доступ до виконання класифікації та керує доступом до даних;
- графічний інтерфейс користувача.

Для забезпечення контролю доступу до даних та розмежування відповідальності введено поняття організації для використання системи багатьма користувачами. Організація є групою користувачів та даних, що їм належать.

Користувачі мають доступ лише до даних своєї організації, що дозволяє безпечно використовувати одну інсталяцію програмного забезпечення декількома незалежними установами, що досягається шляхом розділення прав користувачів. Основні варіанти використання системи представлено на рис. 4.1.

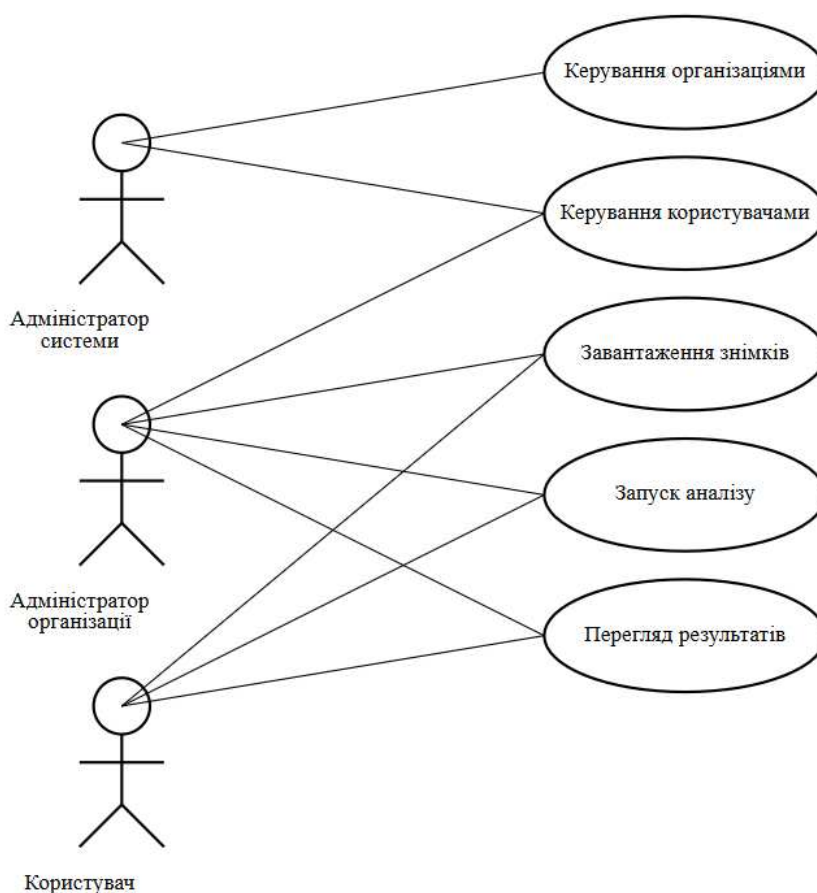


Рисунок 4.1 – Діаграма варіантів використання

Система передбачає три рівні доступу з відповідною ієрархією прав користувачів:

- «Адміністратор системи» – має повноваження щодо керування організаціями та глобального адміністрування користувачів;
- «Адміністратор організації» – має права на керування користувачами в межах своєї організації та доступ до всіх функціональних можливостей системи;
- «Користувач» – має доступ лише до основних операцій: завантаження знімків, запуску аналізу та перегляду результатів у межах своєї організації.

Така ієрархічна структура прав доступу забезпечує належний рівень безпеки даних та ефективно розмежування адміністративних повноважень при збереженні простоти керування системою.

4.1 Компоненти програмного засобу

Для забезпечення функціональності та масштабованості програмного засобу необхідна взаємодія декількох спеціалізованих компонентів, кожен з яких відповідає за окремий аспект роботи системи – від зберігання даних до взаємодії з користувачем.

Розроблена система складається з наступних компонент:

- реляційна база даних (зберігає інформацію про користувачів та метадані аналізованих знімків);
- векторна база даних (виконує пошук найближчих знімків за вкладеними представленнями);
- система керування потоками робіт ;
- сервер ПЗ;
- клієнт ПЗ.

Архітектурне рішення щодо розгортання програмного забезпечення базується на використанні платформи оркестрації контейнерів Kubernetes, що обумовлено як технічними вимогами СКПР на основі запропонованого методу, так і перевагами даної платформи для складних розподілених систем.

Розгортання усіх компонентів системи здійснюється за допомогою декларативних маніфестів Kubernetes, що описують бажаний стан системи. Це забезпечує надійну відтворюваність розгортання та спрощує внесення змін у конфігурацію системи, а також дозволяє гнучко керувати обчислювальними ресурсами відповідно до поточних потреб. Діаграму компонент наведено на рис. 4.2.

Для збереження інформації про користувачів, організації та завантажені знімки було обрано систему керування реляційними базами даних PostgreSQL. Такий вибір обґрунтовано здатністю PostgreSQL до масштабування та надійної роботи з великою кількістю даних.

Для реалізації пошуку схожих випадків за допомогою вкладених представлень було використано векторну базу даних. Векторні бази даних дозволяють зберігати вектори поряд з іншими даними. Для пошуку та видобування інформації, як правило, використовуються один чи декілька алгоритмів апроксимованого пошуку найближчих векторів. Пошук у векторних базах даних здійснюється шляхом надання вектору запиту. Результатом пошуку є n найближчих векторів та прив'язані до них дані

Така будова векторних баз даних значно спрощує масштабування та паралелізацію при виконанні запитів у базу даних, а також надає можливість роботи з багатовимірними та неструктурованими чи нереляційними даними при збереженні ефективного видобування інформації [104].

Роботу векторної бази даних можна розбити на два принципових етапи: індексування та пошук.

У процесі індексування, дані, що необхідно зберегти, векторизуються у вкладене представлення цих даних (у запропонованій системі, що розглядається, даний етап виконується поза базою даних), і отримані вкладені представлення «стискаються». Під час стискання (компресування) наявних даних, БД визначає групи схожих векторів («класи»), для яких обчислюються усереднені репрезентації. Чим ближчими є два вкладені представлення, тим більшою є імовірність їх потрапляння в одну групу.

Так само, етап пошуку починається з векторизації запиту. Отриманий вектор порівнюється з репрезентативними векторами груп-класів з метою ідентифікації класу що має найбільшу схожість до вектору запиту. Серед членів знайденої найближчої групи проводяться почергові порівняння шукаемого вектору з векторами наявних у базі даних. Дані, пов'язані з найближчими з векторів є результатом запиту.

Очевидно, що прямий пошук послідовним порівнянням шуканого вектору з векторами збережених даних є неефективним і тому незастосовним для виконання реальних задач. Відтак, для стискання векторів у групи та пошуку можуть використовуватись різні алгоритми:

- k-найближчих сусідів;
- локальнісно-чутливе хешування (англ. «Locality-Sensitive Hashing») [105]
- ієрархічні прохід-придатні тісні світи (англ. «Hierarchical navigable small world») [106] тощо.

Ієрархічні прохід-придатні тісні світи є широко застосовуваними в сучасних векторних базах даних завдяки гнучкості, придатності до роботи з багатовимірними векторами та швидкості [107].

Зазначені алгоритми працюють з узагальненим поняттям подібності векторів. Відтак, для виконання порівняння можуть бути застосовані різні способи виміру подібності, наприклад (4.1-4.3):

- косинусна подібність (4.1)

$$d_{cos} = \frac{A \cdot B}{\|A\| \|B\|}; \quad (4.1)$$

- евклідова відстань (4.2)

$$d_{euc} = \|A - B\|; \quad (4.2)$$

- скалярний добуток (4.3)

$$d_{scal} = A \cdot B = \sum_{i=1}^n a_i b_i, \quad (4.3)$$

– де A та B є векторами, щодо яких обчислюється подібність, a_i та b_i є елементами даних векторів.

Різні міри подібності мають різну застосовність до різних векторизованих даних. Так, евклідова відстань дозволяє акцентувати на дистанції між векторами, косинусна подібність виражає кут між векторами, а скалярний добуток балансує між обидвома видами схожості.

Відтак, евклідова відстань найбільше підходить для векторів неперервних даних, косинусна подібність – для текстових та багатовимірних, скалярний добуток – для загальної подібності у векторному просторі [104].

Більшість векторних баз даних орієнтовані на роботу з текстом та текст-орієнтований пошук найближчих. Такі БД часто використовуються для рекомендаційних рушіїв, семантичних пошукових систем, ідентифікації аномалій у банківських транзакціях, персоналізації реклами тощо. Такі бази даних виконують векторизацію тексту приховано від користувача. Інтеграція користувацьких засобів побудови вкладених представлень тексту у таких БД виконується напряму у базу даних.

Оскільки вкладені представлення що є ключами векторної БД, обчислюються за допомогою відносно важких нейронних мереж у потоці робіт, для програмного засобу такі бази даних (як, наприклад Qdrant, Chroma тощо) не є застосовними.

Серед векторних баз даних, що дозволяють записувати створені поза базою даних вкладені представлення, найбільш популярними є:

- Pinecone [108];
- Milvus [109];
- Weaviate [110].

Для даного програмного засобу пріоритетними критеріями вибору рішення є відкритість початкового коду, спосіб використання (DBaaS чи локальне розгортання) та здатність до динамічного та швидкого масштабування. Розглянемо існуючі рішення враховуючи дані критерії.

Pinecone є пропрієтарною векторною базою даних, що постачається як хмарна послуга (DBaaS – без можливості запуску на власному хмарному чи фізичному

обладнанні). Завдяки цьому, ця векторна база даних є добре масштабованою та пристосованою до розподілених обчислень, оскільки відповідальність за масштабування, розподілення та керування ресурсами Pinecone бере на себе як надавач хмарних послуг. Однак, використання бази даних за такою моделлю є дорогим та несе ризики через прив'язку до постачальника.

Milvus є векторною базою даних з відкритим початковим кодом. Також, даний програмний засіб є орієнтованим на масштабовану роботу в кластерному режимі у декілька вузлів, та підтримує просте розгортання на Kubernetes за допомогою менеджера шаблонів розгортання Helm. Milvus також допускає вибір різних алгоритмів індексації, що дозволяє оптимізувати баланс між точністю та швидкістю на наявних даних. На противагу, можливості з аналізу даних та інтеграції з іншими базами даних є обмеженими [104]. У кластерному розгортанні, Milvus підтримує динамічне масштабування кількості вузлів та алокацію ресурсів на ті чи інші задачі.

Як і Milvus, Weaviate є векторною базою даних з відкритим кодом та підтримкою власного розгортання та використання поза хмарним середовищем розробника. Однією з переваг Weaviate є модульна архітектура, що дозволяє розширення функціоналу БД конекторами до інших джерел даних та підключення різноманітних моделей машинного навчання [111]. На відміну від Milvus, розширення кластера Weaviate вимагає ручного додавання вузлів та перерозміщення даних між вузлами, що сповільнює масштабування.

Для зберігання вкладених представлень та пошуку найближчих за ними у даному програмному засобі може бути застосоване будь-яке з розглянутих рішень. Milvus було обрано з огляду на можливість локального розгортання та здатність до швидкого масштабування, відкритий початковий код.

Виконання ресурсоємних обчислень здійснюється за допомогою потоків робіт. Керування ними реалізовано за допомогою розробленої СКПР (див. пп. 3.2).

Серверний компонент програмного засобу реалізовано у вигляді HTTP-сервера на базі платформи Node.js та фреймворку NestJS. Графічний інтерфейс користувача реалізовано за допомогою Angular.

Діаграму компонент програмного засобу наведено на (рис 4.2).

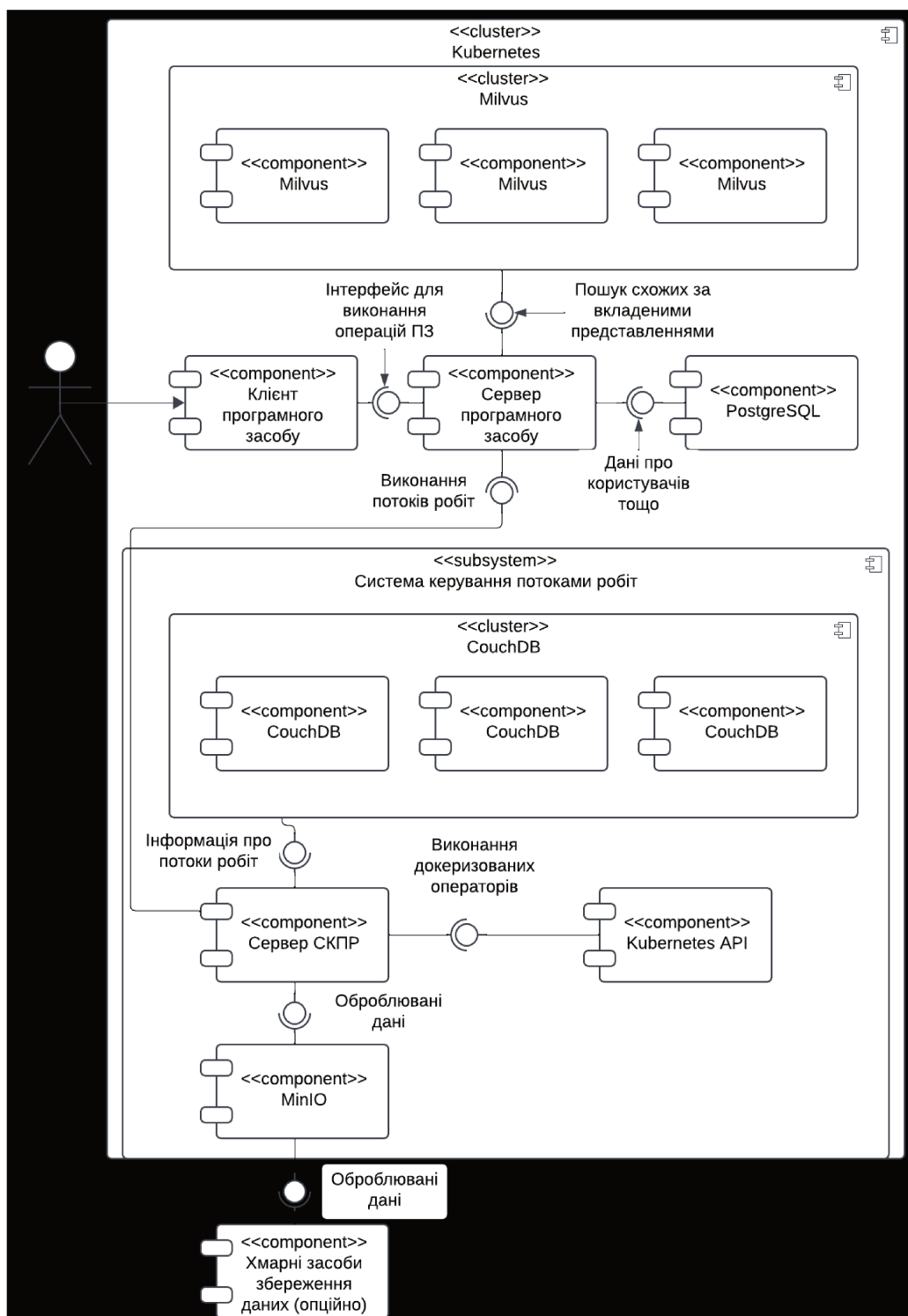


Рисунок 4.2 – Діаграма компонент програмного засобу обробки знімків КТ

Розроблена структура містить описані компоненти та спрямована на мінімізацію часу адаптації при зміні вимог до функціоналу системи.

4.2 Реалізація

Програмний засіб містить реалізацію наступних елементів:

1. оператори для системи керування потоками робіт;
2. сервер програмного засобу;
3. клієнт програмного засобу.

Такий поділ забезпечує чітке розмежування відповідальності між компонентами: оператори виконують обчислювальні задачі в рамках потоків робіт, серверна частина реалізує бізнес-логіку процесу, а клієнтська частина відповідає за взаємодію з користувачем.

Комунікація в системі організована наступним чином:

- Взаємодія між клієнтом та сервером здійснюється через REST API;
- Внутрішньокластерна взаємодія сервера з СКПР та операторами, що запускаються в ній реалізована шляхом HTTP-запитів;
- Оператори ізольовані від прямої взаємодії з іншими компонентами системи, обмін даними відбувається через файлову систему під керуванням СКПР.

Така структура комунікації забезпечує надійну та масштабовану взаємодію між усіма компонентами системи.

4.2.1 Оператори для підсистеми керування потоками робіт

Розроблений програмний засіб приймає та обробляє як тривимірні КТ-знімки так і окремі зрізи. При обробці знімка кожен зріз векторизується та класифікується окремо. Однак інформація про усі зрізи знімка проходить по оператору «разом». Відтак, для кожного завантаженого знімка запускається лише один екземпляр кожного оператора, що дозволяє економити час та ресурси на виділення обчислювального простору для виконання аналізу.

Для виконання класифікації тривимірного знімка КТ необхідно здійснити наступні дії:

- вибрати зрізи, що містять легені;

- перекодувати зображення у формат, що є зручним для обробки нейронною мережею;
- побудувати вкладені представлення зрізів;
- виконати класифікацію за вкладеними представленнями.

Для забезпечення даного процесу визначено наступні види даних (табл. 4.1).

Таблиця 4.1 – Види даних потоку робіт класифікації знімків КТ

| Назва | Умовне позначення | Опис |
|--------------------------------|-------------------|--|
| dicom-chest-ct | DC | Тривимірний знімок КТ у DICOM |
| dicom-chest-ct-single | DCS | Двовимірний зріз КТ у DICOM |
| tiff-chest-ct-filtered | TCF | Зріз(-и) КТ, готові до обробки, у TIFF |
| chest-ct-embeddings | CE | Вкладені представлення зрізу(-ів) КТ |
| chest-ct-classification-result | CCR | Результат класифікації зрізу(-ів) КТ |

Для обробки описаних видів даних було розроблено ряд операторів (табл. 4.2).

Таблиця 4.2 – Оператори, що складають потік робіт класифікації знімків КТ

| Назва | Умовне позначення | Запис | Виконує |
|---------------------------|-------------------|-------------------|--|
| dicom-split-filter | dsf | dsf (DC) => TCF | попередню обробку тривимірних знімків – визначення зрізів, що містять легені та їх перекодування у формат, що сприймається рештою операторів |
| single-dicom-slice-packer | sdsp | sdsp (DCS) => TCF | попередню обробку двовимірних зрізів – кодування у формат, що сприймається рештою операторів |
| resnet-fdn-embed | rfe | rfe (TCF) => CE | побудову вкладених представлень для кожного зрізу знімка КТ за допомогою нейронної мережі |
| neigh-embed-classify | nec | nec (CE) => CCR | класифікацію захворювання на основі ВП зрізів за допомогою класифікатора на основі методу k-найближчих сусідів |
| randfor-embed-classify | rec | rec (CE) => CCR | класифікацію захворювання на основі ВП зрізів за допомогою класифікатора на основі методу випадкового лісу |
| svc-embed-classify | sec | sec (CE) => CCR | класифікацію захворювання на основі ВП зрізів за допомогою класифікатора на основі методу опорних векторів |

З даних операторів складається наступний потік робіт (рис. 4.3).

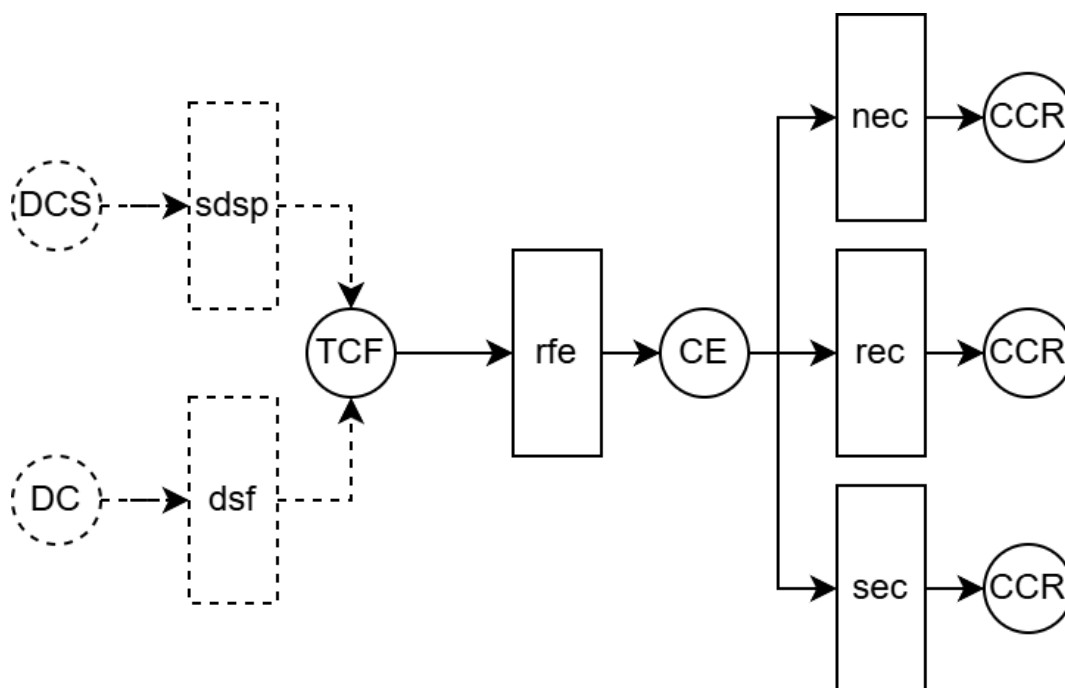


Рисунок 4.3 – Потік робіт для обробки знімка КТ

Потік робіт може прийняти на вхід як тривимірний знімок, так і двовимірний зріз. У випадку завантаження повного знімка, він розділяється на окремі зрізи. З наданих зрізів обираються такі, що містять зображення легень (див. пп. 2.1), та передаються для подальших кроків у вигляді zip-архіву. Натомість якщо буде завантажено окремий зріз, він передається для подальших кроків у вигляді zip-архіву з одного файлу.

Після отримання готових до обробки файлів над зрізами КТ здійснюється побудова вкладених представлень за допомогою нейронних мереж. Отримані ВП зрізів використовуються для класифікації за допомогою трьох алгоритмів машинного навчання:

- k-найближчих сусідів;
- випадкового лісу;
- методу опорних векторів.

Усі оператори класифікації за вкладеними представленнями видають результат у вигляді JSON однакової структури. У результатах класифікації (вид даних chest-ct-classification-result) міститься масив, у якому для кожного вкладеного представлення наведено результат у вигляді рядку:

```
{
  prediction_strings: [..., "covid", "normal", ...]
}
```

Таким чином, розроблений набір операторів для СКПР дозволяє розширення програмного засобу:

- новими форматами зберігання даних;
- новими методами аналізу знімків КТ для діагностики;
- іншими медичними даними та відповідними їм методами аналізу.

При цьому, для здійснення таких модифікацій достатнім є лише додавання нових чи зміна наявних операторів СКПР без внесення змін в будь-які інші компоненти програмного засобу. Також, використання СКПР на основі динамічної побудови графу потоку робіт дозволяє істотно знизити час на запровадження таких змін у порівнянні з традиційними СКПР чи реалізацією у вигляді мікросервісів.

4.2.2 Серверна частина програмного засобу

Сервер програмного засобу складається з наступних модулів:

1. Analysis – модуль, що відповідає за:
 - a. завантаження даних;
 - b. ініціацію потоків робіт у СКПР;
 - c. обробку результатів аналізу;
2. User – модуль, що відповідає за керування користувачами;
3. Authentication – модуль, що відповідає за автентифікацію користувачів.

Аналіз знімків КТ виконується шляхом завантаження вхідних даних до СКПР та запуску потоку робіт.

Для отримання результат роботи ПР проводиться опитування сервера програмного засобу про готовність результату обчислень. У перший запит про готовність результату коли буде виявлено що потік робіт який виконує аналіз даного знімка завершено виконується обробка результатів ПР – для кожного алгоритму

окремо проводиться агрегація результатів класифікації по зрізах для визначення остаточного рішення.

Рішенням методу обирається результат класифікації, який збігається між найбільшою кількістю зрізів. Тобто, якщо метод з (у якості прикладу) 15 зрізів знімка на 10 зрізах показує результат «пневмонія, спричинена COVID-19», та на 5 зрізах показує «здорові легені», рішенням класифікації знімка методом вважається «пневмонія, спричинена COVID-19». Аналогічно до рішень окремих методів було втілено загальне рішення аналізу – результат найбільшої кількості методів стає загальним рішенням аналізу.

При отриманні результату ПР обчислені вкладені представлення зберігаються у векторну базу даних із зазначенням двох ідентифікаторів:

- організації, яка виконує аналіз;
- виконаного аналізу.

Для виконання пошуку найближчих видобуваються ВП зрізів обробленого знімка та виконується пошуковий запит у векторну базу даних за даними векторами. Запит конфігурується умовою, що результати пошуку мають містити виключно результати з організації, яка є власником даного знімка, та не мають містити результатів, що відповідають векторам даного знімка.

Оскільки векторна база даних містить багато показників на один і той самий аналіз, і, відповідно, одне і те саме КТ-зображення – результати пошуку, що відповідають одному і тому самому знімку, збираються в окремі масиви. З цих масивів обираються вкладені представлення зрізів з найбільшою оцінкою близькості до шуканих векторів. Результатами пошуку є аналізи (та їх знімки КТ), відсортовані за визначеним найближчим з знайдених знімків.

Виходячи з вимоги до мультитенантності та контролю доступу до даних було втілено систему організацій та керування доступом на основі ролей. Реєстрація користувачів системи є закритою: для отримання доступу користувач мусить бути внесений «адміністратором організації» чи «адміністратором програмного засобу».

Кожен користувач має належати до однієї організації. У випадку приналежності персони до декількох організацій, у кожній вона отримує окремий обліковий запис та одну з трьох ролей:

- користувач;
- адміністратор організації;
- адміністратор ПЗ.

«Користувач» та «адміністратор організації» мають доступ до завантажених організацією даних та проведених аналізів на основі цих даних.

«Адміністратор організації» має доступ до додаткової сторінки управління користувачами, за допомогою якої може вносити та видаляти користувачів у організації, до якої належить.

«Адміністратор ПЗ» має доступ до окремого клієнтського засобу, який надає можливість створювати, переглядати, редагувати та видаляти організації, переглядати, створювати, редагувати, видаляти користувачів, а також переміщувати їх між організаціями.

Користувач може бути зареєстрованим виключно після внесення його електронної пошти до бази даних з прив'язкою до організації. Паролі зберігаються у хешованому вигляді (з використанням алгоритму bcrypt [112] з фактором складності, що дорівнює 12).

Під час входу у систему пароль порівнюється з хешем, і при рівності хешів користувачу видається JWT-токен з ідентифікатором користувача у якості HTTP-only cookie (зберігаються браузером, надаються серверу при запитах, не доступні з JavaScript-коду фронтенду та плагінів браузера) дійсний протягом однієї доби.

Під час будь-якої дії користувача всередині програмного забезпечення перевіряється наявність у користувача дійсного JWT-токену та право користувача на доступ до запитуваного ресурсу: користувач може зчитувати та модифікувати аналізи виключно в межах організації, до якої прив'язаний, доступ до списку користувачів організації обмежений роллю.

Доступ до сторінки адміністрування ПЗ для користувачів-адміністраторів ПЗ перевіряється за збігом з наявними у базі поштової скриньки та пароллю, а також наявності відповідної ролі у користувача.

4.2.3 Клієнтська частина програмного засобу

Клієнтська частина програмного засобу складається з двох веб-застосунків:

- інтерфейсу адміністратора системи, реалізованого за допомогою фреймворку AdminJS;
- інтерфейсу користувачів та адміністраторів організацій, реалізованого на базі фреймворку Angular з використанням компонентної бібліотеки Angular Material.

Структура застосунку включає сторінки авторизації, керування користувачами та аналізу.

Сторінка аналізу надає наступний набір інструментів:

- завантаження вхідних даних;
- запуск процесу аналізу;
- перегляд результатів класифікації;
- пошук та перегляд схожих випадків за допомогою векторної бази даних.

Взаємодія з серверною частиною реалізована через REST API з використанням вбудованого в Angular HTTP-клієнта. Клієнтська частина програмного засобу використовує CRUD-операції для керування аналізами та користувачами (для адміністраторів організацій), а також спеціалізовані операції для роботи з даними аналізу, запуску обробки даних та пошуку схожих випадків.

Для забезпечення консистентного користувацького досвіду застосовано бібліотечні компоненти Angular Material, що надають уніфікований зовнішній вигляд та поведінку елементів інтерфейсу відповідно до сучасних стандартів проектування веб-інтерфейсів.

4.3 Пошук найближчих за допомогою векторної бази даних

Однією з переваг використання вкладених представлень є можливість застосування векторних баз даних для пошуку схожих зрізів за близькістю ВП. Даний функціонал може слугувати допоміжним інструментом лікаря, реалізуючи можливість пошуку пацієнтів зі схожим ураженням легень.

Розглянемо приклад пошуку векторною базою даних за вкладеними представленнями, згенерованими нейронною мережею (див. розділ 2).

Для експериментів використовувалась база даних, що містить вкладенні представлення 1501 зрізу з 27 знімків КТ (9 COVID-19, 9 позагоспітальна пневмонія, 9 здорові легені) датасету COVID-СТ-MD. Для пошуку використовувалась вбудована порівняльна міра Milvus – евклідова відстань (4.2) за вкладеним представленням наступного зрізу (рис. 4.4).



Рисунок 4.4 – Зріз IM0046 знімку covid114

Результатом пошуку 5 найближчих були наступні зрізи (рис. 4.5).

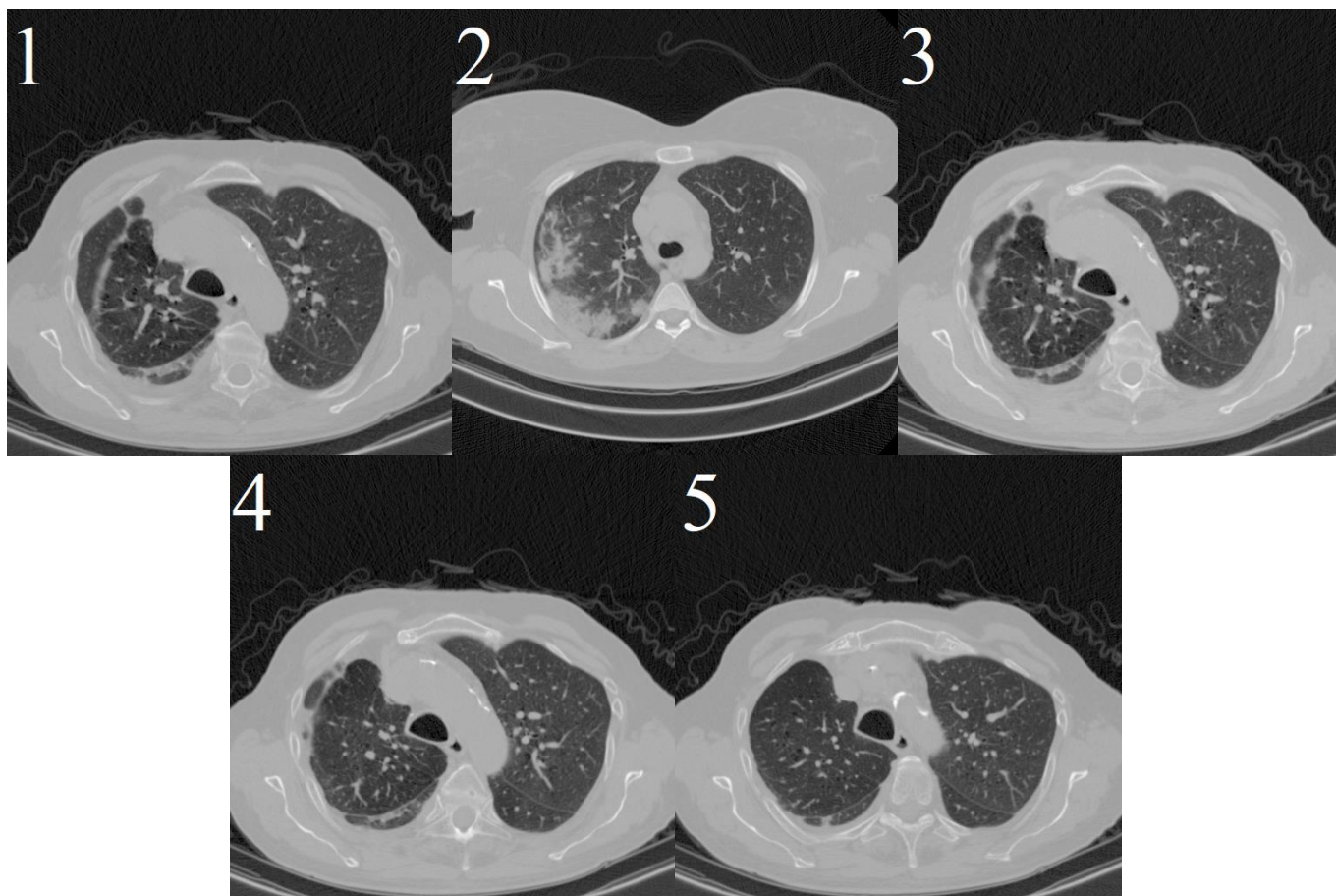


Рисунок 4.5 – Результати пошуку найближчих (за евклідовою відстанню)

Евклідова відстань для знайдених зрізів наведена у табл. 4.3.

Таблиця 4.3 – Результати пошуку найближчих (за евклідовою відстанню)

| Номер результату | Назва знімку | Назва зрізу | Евклідова відстань |
|------------------|--------------|-------------|--------------------|
| 1 | covid110 | IM0038 | 2 148 603,5 |
| 2 | covid115 | IM0046 | 2 163 308,75 |
| 3 | covid110 | IM0037 | 2 177 687,5 |
| 4 | covid110 | IM0036 | 2 210 008,75 |
| 5 | covid110 | IM0032 | 2 373 189,75 |

Оскільки експериментальна база даних не містить значної кількості знімків, додатково було обчислено відстані між ВП запиту та ВП усіх інших зрізів присутніх у базі (загальною кількістю 1501), та відібрано 5 зрізів з найбільш далекими ВП (рис. 4.6).

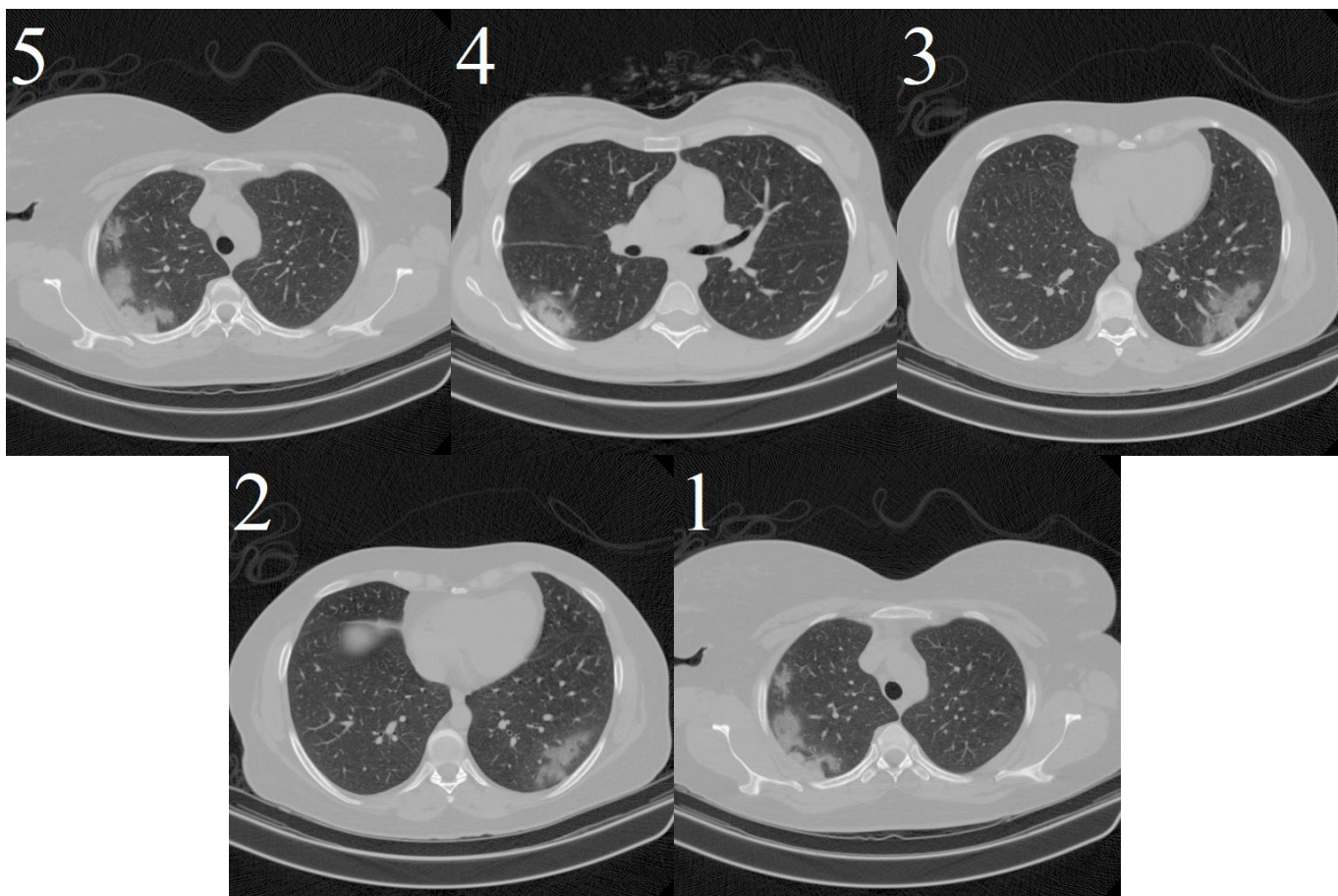


Рисунок 4.6 – Результати пошуку найдальших (за евклідовою відстанню)

Евклідова відстань для знайдених зрізів наведена у табл. 4.4.

Таблиця 4.4 – Результати пошуку найдальших (за евклідовою відстанню)

| Номер результату | Назва знімку | Назва зрізу | Евклідова відстань |
|------------------|--------------|-------------|--------------------|
| 5 | covid115 | IM0038 | 10 587 443 |
| 4 | covid111 | IM0092 | 10 840 212 |
| 3 | covid115 | IM0094 | 11 268 558 |
| 2 | covid115 | IM0036 | 11 430 684 |
| 1 | covid115 | IM0037 | 13 252 108 |

Як можна побачити з табл. 4.3-4.4 відношення евклідових відстаней між найближчими та найдальшими зображеннями становить майже 5 разів, що є достатнім для їх розділення.

Нормалізовані ВП усіх відібраних зрізів наведено на рис. 4.7.

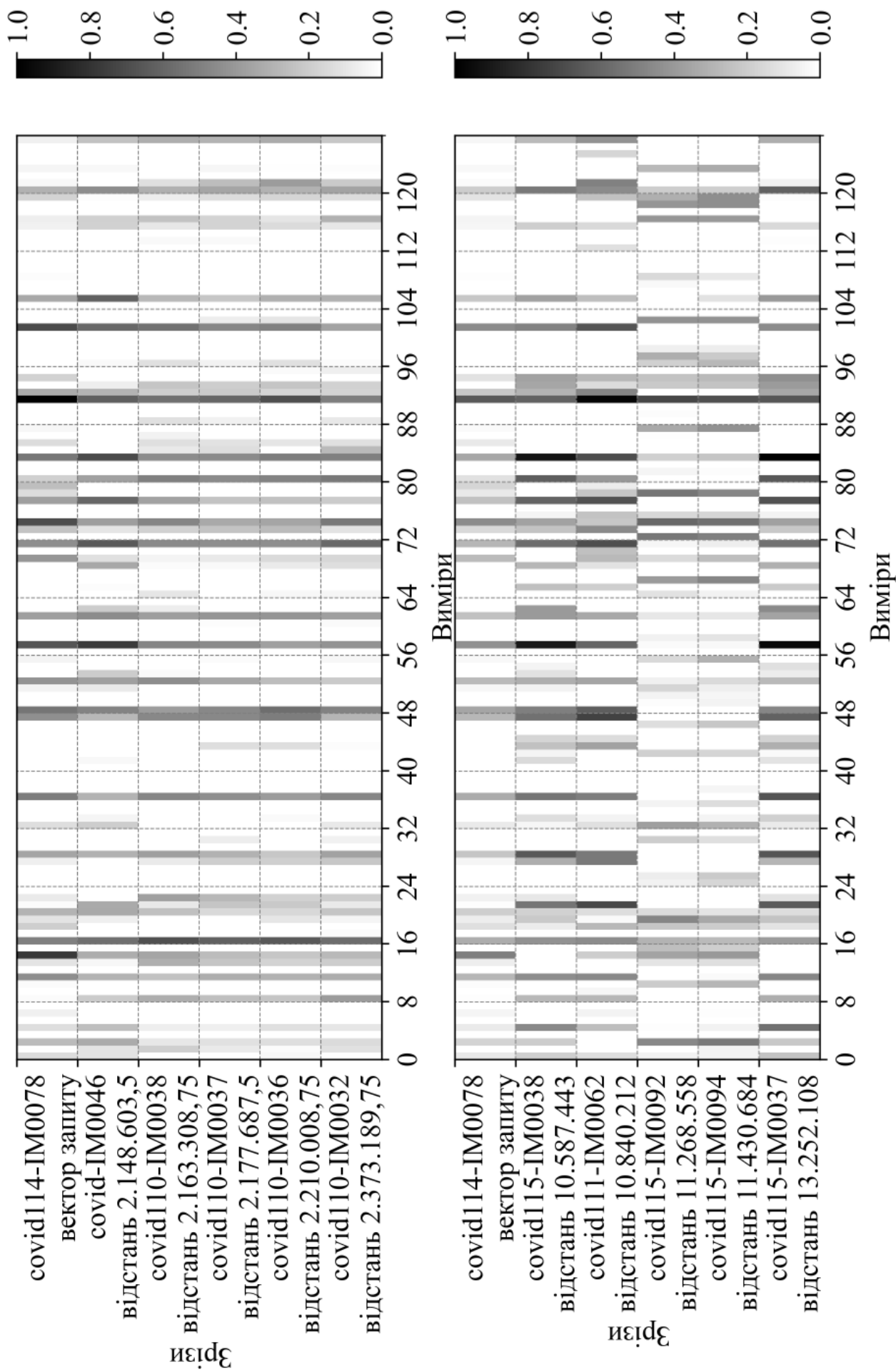


Рисунок 4.7 – Нормалізовані вкладки представлення відібраних зрізів

Як можна побачити з рис. 4.7, вкладені представлення для одного типу зображень (зрізів КТ) мають помітні відмінності. Так, в октетах 40-48 та 80-88 можна побачити різницю між найближчими та найдальшими за евклідовою відстанню зображеннями.

Пошук за схожістю вкладених представлень знімків КТ може бути корисним при оцінці ураження легень, прийнятті рішень та прогнозуванні стану пацієнта.

4.4 Висновки до розділу

У розділі розглянуто реалізацію програмного забезпечення аналізу знімків КТ для діагностики.

Розроблено архітектуру системи, що складається з:

- підсистеми керування потоками робіт;
- систем керування базами даних;
- серверу системи;
- операторів, що реалізують побудову вкладених представлень знімків КТ (у складі СКПР);
- операторів, що виконують класифікацію за вкладеними представленнями (у складі СКПР);
- інтерфейсів користувачів за ролями.

Описано особливості реалізації системи з урахуванням поняття організацій та розділенням прав доступу. Наведено приклади пошуку вкладених представлень за допомогою векторної бази даних.

Розроблене програмне забезпечення покриває нагальні задачі, пов'язані з діагностикою легеневих захворювань шляхом аналізу знімків КТ. Подальший розвиток системи може бути пов'язаний як з розширенням набору легеневих захворювань, що підтримуються системою (туберкульоз, новоутворення тощо), так і з введенням підтримки принципово інших даних для діагностування захворювань інших частин тіла (КТ інших органів, МРТ тощо).

ВИСНОВКИ

У дисертаційній роботі вирішено наукове завдання зменшення часу адаптації програмного забезпечення аналізу медичних зображень для діагностики на основі алгоритмів машинного навчання

Основні результати виконаної роботи:

1. Проведено аналіз літературних джерел, щодо методів аналізу знімків КТ для діагностики захворювань.
2. Проведено аналіз можливостей існуючих систем керування потоками робіт.
3. Запропоновано модифікацію топології існуючої мережі бінарної класифікації зображень для мультикласової класифікації та доведено можливість розширення новими класами без істотної втрати точності.
4. Запропоновано та розроблено метод класифікації зображень на основі вкладених представлень, який забезпечує можливість пришвидшеного донавчання нейронної мережі для підтримки нових класів без зміни її топології, що зменшує час на адаптацію програмного забезпечення аналізу медичних зображень для діагностики. Метод використано у задачі аналізу знімків КТ з досягненням точності класифікації у 98.7%.
5. Розроблено метод динамічного конструювання потоків робіт, що виконує побудову графу під час виконання, та дозволяє уникнути побудови статичних потоків робіт вручну, що зменшує час на розробку та модифікацію систем, які використовують потоки робіт.
6. Розроблено програмну систему керування потоками робіт, яка використовує кластерні обчислення на основі Kubernetes та підтримує горизонтальне масштабування. Розроблено математичну модель обрахунку часу виконання потоку робіт. Досліджено швидкодію алгоритму в ізоляції та системи керування потоками робіт, та доведено їх застосовність до практичних задач.
7. Розроблено математичні моделі для оцінки часу модифікації класифікатора та побудови графу потоку робіт, що враховують складові часу даних операцій. За допомогою розроблених моделей доведено практичну

застосовність розроблених методів . При використанні реальних параметрів при розробці та запровадженні змін у програмне забезпечення доведено зменшення часу на його адаптацію.

8. Розроблено програмне забезпечення аналізу знімків КТ для діагностики захворювань на основі розглянутого методу класифікації та запропонованої системи керування потоками робіт з можливістю використання декількома організаціями.

Розроблені в дисертації програмні засоби можуть використовуватись як окремо (СКПР, ПЗ аналізу знімків КТ для діагностики) так і в складі інших систем (наприклад, як складові систем підтримки прийняття рішень).

Таким чином, поставлені у даній дисертаційній роботі завдання виконано повністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] R. A. Weiss and A. J. McMichael, “Social and environmental risk factors in the emergence of infectious diseases.,” *Nat Med*, vol. 10, no. 12 Suppl, pp. S70-6, Dec. 2004, doi: 10.1038/nm1150.
- [2] J. Poorolajal, “The global pandemics are getting more frequent and severe.,” *J Res Health Sci*, vol. 21, no. 1, p. e00502, Jan. 2021, doi: 10.34172/jrhs.2021.40.
- [3] C. D. Kelly-Cirino *et al.*, “Importance of diagnostics in epidemic and pandemic preparedness.,” *BMJ Glob Health*, vol. 4, no. Suppl 2, p. e001179, 2019, doi: 10.1136/bmjgh-2018-001179.
- [4] M. D. Perkins *et al.*, “Diagnostic preparedness for infectious disease outbreaks,” *The Lancet*, vol. 390, no. 10108, pp. 2211–2214, Nov. 2017, doi: 10.1016/S0140-6736(17)31224-2.
- [5] R. Lowe *et al.*, “The Zika Virus Epidemic in Brazil: From Discovery to Future Implications.,” *Int J Environ Res Public Health*, vol. 15, no. 1, Jan. 2018, doi: 10.3390/ijerph15010096.
- [6] “New tool indicates woeful diagnostic readiness for pandemic prevention, preparedness and response,” FIND. Accessed: Sep. 19, 2024. [Online]. Available: <https://www.finddx.org/publications-and-statements/press-release/new-tool-indicates-woeful-diagnostic-readiness-for-pandemic-prevention-preparedness-and-response/>
- [7] F. Smilianets and O. Finogenov, “Multi-Class Classification of Pulmonary Diseases Using Computer Tomography Images,” *Adaptive Systems of Automatic Control*, vol. 2, no. 43, pp. 78–83, Dec. 2023, doi: 10.20535/1560-8956.43.2023.292255.
- [8] Ф. Смілянець та О. Фіногенов, “Мультикласова класифікація легеневих захворювань за допомогою знімків комп’ютерної томографії,” у *Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023) : матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів, присвячених 125-й річниці КПІ ім. Ігоря Сікорського, 2023*, с. 28–30.

- [9] F. Smilianets, “Application of embeddings for multi-class classification with optional extendability,” *Adaptive Systems of Automatic Control*, vol. 2, no. 45, pp. 186–193, Oct. 2024, doi: 10.20535/1560-8956.45.2024.313198.
- [10] F. Smilianets, “Application of Transfer Learning for enhanced pulmonary disease detection via CT image embeddings,” *Adaptive Systems of Automatic Control*, vol. 1, no. 44, pp. 24–29, 2024, doi: 10.20535/1560-8956.44.2024.302198.
- [11] Ф. Смілянець та О. Фіногенов, “Методи та програмне забезпечення обробки знімків комп’ютерної томографії для автоматизації діагностики захворювань,” у *Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2024): матеріали VI Міжнародної науково-практичної конференції молодих вчених та студентів*, 2024, с. 124–127.
- [12] Ф. Смілянець та О. Фіногенов, “Порівняння алгоритмів класифікації з використанням вкладених представлень знімків КТ,” у *Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2024): матеріали VII Міжнародної науково-практичної конференції молодих вчених та студентів*, 2024, с. 96–100.
- [13] F. Smilianets and O. Finogenov, “Running a workflow without workflows: a basic algorithm for dynamically constructing and traversing an implied directed acyclic graph in a non-deterministic environment,” *Informatyka, Automatyka, Pomiarы w Gospodarce i Ochronie Środowiska*, vol. 14, no. 1, pp. 115–118, Mar. 2024, doi: 10.35784/iapgos.5858.
- [14] F. A. Smilianets and O. D. Finogenov, “Review of disease identification methods based on computed tomography imagery,” *Ukrainian Journal of Information Technology*, vol. 6, no. 1, pp. 95–100, 2024, doi: 10.23939/ujit2024.01.095.
- [15] Ф. Смілянець та О. Фіногенов, “Огляд методів ідентифікації захворювань на основі знімків комп’ютерної томографії,” у *XII Міжнародна науково-практична конференція «Комплексне забезпечення якості технологічних процесів та систем»*, 2022, с. 220–221.

- [16] P. Sajda, "Machine Learning for Detection and Diagnosis of Disease," *Annu Rev Biomed Eng*, vol. 8, no. Volume 8, 2006, pp. 537–565, 2006, doi: <https://doi.org/10.1146/annurev.bioeng.8.061505.095802>.
- [17] Y. LeCun *et al.*, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., Morgan-Kaufmann, 1989. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf
- [18] T. Zrimec and J. Wong, "Methods for Automatic Honeycombing Detection in HRCT images of the Lung," in *11th Mediterranean Conference on Medical and Biomedical Engineering and Computing 2007*, T. Jarm, P. Kramar, and A. Zupanic, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 830–833. doi: 10.1007/978-3-540-73044-6_215.
- [19] J. Ye, Y. Sun, S. Wang, L. Gu, L. Qian, and J. Xu, "Multi-Phase CT Image Based Hepatic Lesion Diagnosis by SVM," in *2009 2nd International Conference on Biomedical Engineering and Informatics*, 2009, pp. 1–5. doi: 10.1109/BMEI.2009.5304774.
- [20] T. Gong *et al.*, "Classification of CT Brain Images of Head Trauma," in *Pattern Recognition in Bioinformatics*, J. C. Rajapakse, B. Schmidt, and G. Volkert, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 401–408. doi: 10.1007/978-3-540-75286-8_38.
- [21] P. Lakhani and B. Sundaram, "Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks," *Radiology*, vol. 284, no. 2, pp. 574–582, 2017, doi: 10.1148/radiol.2017162326.
- [22] G. Carneiro, L. Oakden-Rayner, A. P. Bradley, J. Nascimento, and L. Palmer, "Automated 5-year mortality prediction using deep learning and radiomics features from chest computed tomography," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, 2017, pp. 130–134. doi: 10.1109/ISBI.2017.7950485.

- [23] X. Zhou, R. Takayama, S. Wang, T. Hara, and H. Fujita, “Deep learning of the sectional appearances of 3D CT images for anatomical structure segmentation based on an FCN voting method,” *Med Phys*, vol. 44, no. 10, pp. 5221–5233, Oct. 2017, doi: 10.1002/mp.12480.
- [24] Y. Xie *et al.*, “Knowledge-based Collaborative Deep Learning for Benign-Malignant Lung Nodule Classification on Chest CT,” *IEEE Trans Med Imaging*, vol. 38, no. 4, pp. 991–1004, 2019, doi: 10.1109/TMI.2018.2876510.
- [25] J. G. Nam *et al.*, “Development and Validation of Deep Learning-based Automatic Detection Algorithm for Malignant Pulmonary Nodules on Chest Radiographs,” *Radiology*, vol. 290, no. 1, pp. 218–228, 2019, doi: 10.1148/radiol.2018180237.
- [26] S. R. S., J. George, S. Skaria, and V. V V., “Using YOLO based deep learning network for real time detection and localization of lung nodules from low dose CT scans,” in *Medical Imaging 2018: Computer-Aided Diagnosis*, N. Petrick and K. Mori, Eds., SPIE, 2018, p. 105751I. doi: 10.1117/12.2293699.
- [27] J. H. and D. S. R. and K. M. K. and P. P. V. and Z. B. and N. C. and S. R. and K. R. D. and L. Q. Li Xiang and Thrall, “Deep learning-enabled system for rapid pneumothorax screening on chest CT,” *Eur J Radiol*, vol. 120, Nov. 2019, doi: 10.1016/j.ejrad.2019.108692.
- [28] D. Ardila *et al.*, “End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography,” *Nat Med*, vol. 25, no. 6, pp. 954–961, 2019, doi: 10.1038/s41591-019-0447-x.
- [29] S. Park *et al.*, “Application of deep learning-based computer-aided detection system: detecting pneumothorax on chest radiograph after biopsy,” *Eur Radiol*, vol. 29, no. 10, pp. 5341–5348, 2019, doi: 10.1007/s00330-019-06130-x.
- [30] N. Nasrullah, J. Sang, M. S. Alam, M. Mateen, B. Cai, and H. Hu, “Automated Lung Nodule Detection and Classification Using Deep Learning Combined with Multiple Strategies,” *Sensors*, vol. 19, no. 17, 2019, doi: 10.3390/s19173722.

- [31] S. M. Humphries *et al.*, “Deep Learning Enables Automatic Classification of Emphysema Pattern at CT,” *Radiology*, vol. 294, no. 2, pp. 434–444, 2020, doi: 10.1148/radiol.2019191022.
- [32] A. Masood *et al.*, “Cloud-Based Automated Clinical Decision Support System for Detection and Diagnosis of Lung Cancer in Chest CT,” *IEEE J Transl Eng Health Med*, vol. 8, pp. 1–13, 2020, doi: 10.1109/JTEHM.2019.2955458.
- [33] X. Wang *et al.*, “A Weakly-Supervised Framework for COVID-19 Classification and Lesion Localization From Chest CT,” *IEEE Trans Med Imaging*, vol. 39, no. 8, pp. 2615–2625, 2020, doi: 10.1109/TMI.2020.2995965.
- [34] H. Ko *et al.*, “COVID-19 Pneumonia Diagnosis Using a Simple 2D Deep Learning Framework With a Single Chest CT Image: Model Development and Validation,” *J Med Internet Res*, vol. 22, no. 6, p. e19569, Jun. 2020, doi: 10.2196/19569.
- [35] X. Yang, X. He, J. Zhao, Y. Zhang, S. Zhang, and P. Xie, “COVID-CT-Dataset: A CT Scan Dataset about COVID-19,” 2020. doi: 10.48550/arXiv.2003.13865.
- [36] X. He *et al.*, “Sample-Efficient Deep Learning for COVID-19 Diagnosis Based on CT Scans,” *medRxiv*, 2020, doi: 10.1101/2020.04.13.20063941.
- [37] O. Gozes, M. Frid-Adar, N. Sagie, H. Zhang, W. Ji, and H. Greenspan, “Coronavirus Detection and Analysis on Chest CT with Deep Learning,” 2020, doi: 10.48550/arXiv.2004.02640.
- [38] Q. Ni *et al.*, “A deep learning approach to characterize 2019 coronavirus disease (COVID-19) pneumonia in chest CT images,” *Eur Radiol*, vol. 30, no. 12, pp. 6517–6527, Dec. 2020, doi: 10.1007/s00330-020-07044-9.
- [39] A. Bhandary *et al.*, “Deep-learning framework to detect lung abnormality – A study with chest X-Ray and lung CT scan images,” *Pattern Recognit Lett*, vol. 129, pp. 271–278, 2020, doi: <https://doi.org/10.1016/j.patrec.2019.11.013>.
- [40] M. Fu *et al.*, “Deep Learning-Based Recognizing COVID-19 and other Common Infectious Diseases of the Lung by Chest CT Scan Images,” *medRxiv*, p. 2020.03.28.20046045, Jan. 2020, doi: 10.1101/2020.03.28.20046045.

- [41] D. Javor, H. Kaplan, A. Kaplan, S. B. Puchner, C. Krestan, and P. Baltzer, “Deep learning analysis provides accurate COVID-19 diagnosis on chest computed tomography,” *Eur J Radiol*, vol. 133, Dec. 2020, doi: 10.1016/j.ejrad.2020.109402.
- [42] X. He *et al.*, “Benchmarking Deep Learning Models and Automated Model Design for COVID-19 Detection with Chest CT Scans,” *medRxiv*, p. 2020.06.08.20125963, Jan. 2021, doi: 10.1101/2020.06.08.20125963.
- [43] E. Matsuyama, “A Deep Learning Interpretable Model for Novel Coronavirus Disease (COVID-19) Screening with Chest CT Images,” *J Biomed Sci Eng*, vol. 13, no. 07, pp. 140–152, 2020, doi: 10.4236/jbise.2020.137014.
- [44] X. Li, Y. Zhou, P. Du, G. Lang, M. Xu, and W. Wu, “A deep learning system that generates quantitative CT reports for diagnosing pulmonary Tuberculosis,” *Applied Intelligence*, vol. 51, no. 6, pp. 4082–4093, 2021, doi: 10.1007/s10489-020-02051-1.
- [45] T. Anwar and S. Zakir, “Deep learning based diagnosis of COVID-19 using chest CT-scan images,” in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 2020, pp. 1–5. doi: 10.1109/INMIC50486.2020.9318212.
- [46] M. Fontanellaz *et al.*, “A Deep-Learning Diagnostic Support System for the Detection of COVID-19 Using Chest Radiographs,” *Invest Radiol*, vol. 56, no. 6, pp. 348–356, Jun. 2021, doi: 10.1097/RLI.0000000000000748.
- [47] M. Yousefzadeh *et al.*, “ai-corona: Radiologist-assistant deep learning framework for COVID-19 diagnosis in chest CT scans,” *PLoS One*, vol. 16, no. 5, p. e0250952, May 2021, doi: 10.1371/journal.pone.0250952.
- [48] H. Alshazly, C. Linse, E. Barth, and T. Martinetz, “Explainable COVID-19 Detection Using Chest CT Scans and Deep Learning,” *Sensors*, vol. 21, no. 2, 2021, doi: 10.3390/s21020455.
- [49] Z. Zhu *et al.*, “Classification of COVID-19 by Compressed Chest CT Image through Deep Learning on a Large Patients Cohort,” *Interdiscip Sci*, vol. 13, no. 1, pp. 73–82, 2021, doi: 10.1007/s12539-020-00408-1.
- [50] F. Pan *et al.*, “A novel deep learning-based quantification of serial chest computed tomography in Coronavirus Disease 2019 (COVID-19),” *Sci Rep*, vol. 11, no. 1, p. 417, 2021, doi: 10.1038/s41598-020-80261-w.

- [51] S. Biswas, S. Chatterjee, A. Majee, S. Sen, F. Schwenker, and R. Sarkar, "Prediction of COVID-19 from Chest CT Images Using an Ensemble of Deep Learning Models," *Applied Sciences*, vol. 11, no. 15, 2021, doi: 10.3390/app11157004.
- [52] W. Zouch *et al.*, "Detection of COVID-19 from CT and Chest X-ray Images Using Deep Learning Models," *Ann Biomed Eng*, vol. 50, no. 7, pp. 825–835, 2022, doi: 10.1007/s10439-022-02958-5.
- [53] A. Ortiz *et al.*, "Effective deep learning approaches for predicting COVID-19 outcomes from chest computed tomography volumes," *Sci Rep*, vol. 12, no. 1, p. 1716, 2022, doi: 10.1038/s41598-022-05532-0.
- [54] N. A. Baghdadi, A. Malki, S. F. Abdelaliem, H. Magdy Balaha, M. Badawy, and M. Elhosseini, "An automated diagnosis and classification of COVID-19 from chest CT images using a transfer learning-based convolutional neural network," *Comput Biol Med*, vol. 144, p. 105383, 2022, doi: <https://doi.org/10.1016/j.compbiomed.2022.105383>.
- [55] F. Sadik, A. G. Dastider, M. R. Subah, T. Mahmud, and S. A. Fattah, "A dual-stage deep convolutional neural network for automatic diagnosis of COVID-19 and pneumonia from chest CT images," *Comput Biol Med*, vol. 149, p. 105806, 2022, doi: <https://doi.org/10.1016/j.compbiomed.2022.105806>.
- [56] S. Asif, Y. Wenhui, S. Jinhai, Z. Waheed, Y. Yueyang, and H. Jin, "CVD19-Net: An Automated Deep Learning Model for COVID-19 Screening using Chest CT Images," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2022, pp. 2049–2058. doi: 10.1109/BIBM55620.2022.9995590.
- [57] K. Gupta and V. Bajaj, "Deep learning models-based CT-scan image classification for automated screening of COVID-19," *Biomed Signal Process Control*, vol. 80, p. 104268, 2023, doi: <https://doi.org/10.1016/j.bspc.2022.104268>.
- [58] O. Attallah, "RADIC:A tool for diagnosing COVID-19 from chest CT and X-ray scans using deep learning and quad-radiomics," *Chemometrics and Intelligent Laboratory Systems*, vol. 233, p. 104750, 2023, doi: <https://doi.org/10.1016/j.chemolab.2022.104750>.

- [59] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognit*, vol. 30, no. 7, pp. 1145–1159, 1997, doi: [https://doi.org/10.1016/S0031-3203\(96\)00142-2](https://doi.org/10.1016/S0031-3203(96)00142-2).
- [60] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788. doi: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [61] X. He *et al.*, “Sample-Efficient Deep Learning for COVID-19 Diagnosis Based on CT Scans,” *medRxiv*, 2020, doi: [10.1101/2020.04.13.20063941](https://doi.org/10.1101/2020.04.13.20063941).
- [62] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” 2018. doi: [10.48550/arXiv.1608.06993](https://doi.org/10.48550/arXiv.1608.06993).
- [63] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2015. doi: [10.48550/arXiv.1512.03385](https://doi.org/10.48550/arXiv.1512.03385).
- [64] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2015. doi: [10.48550/arXiv.1409.1556](https://doi.org/10.48550/arXiv.1409.1556).
- [65] Y. Guo, Y. Li, R. Feris, L. Wang, and T. Rosing, “Depthwise Convolution is All You Need for Learning Multiple Visual Domains,” 2019. doi: [10.48550/arXiv.1902.00927](https://doi.org/10.48550/arXiv.1902.00927).
- [66] M. Tan and Q. V Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” 2020. doi: [10.48550/arXiv.1905.11946](https://doi.org/10.48550/arXiv.1905.11946).
- [67] X. He *et al.*, “Sample-Efficient Deep Learning for COVID-19 Diagnosis Based on CT Scans,” *medRxiv*, p. 2020.04.13.20063941, Jan. 2020, doi: [10.1101/2020.04.13.20063941](https://doi.org/10.1101/2020.04.13.20063941).
- [68] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment.,” *Linux journal*, vol. 2014, no. 239, 2014, Accessed: Sep. 29, 2024. [Online]. Available: <https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment>
- [69] P. Amstutz, M. Mikheev, M. R. Crusoe, N. Tijanić, S. Lampa, and et al., “Existing Workflow systems.,” 2024. [Online]. Available: <https://s.apache.org/existing-workflow-systems>

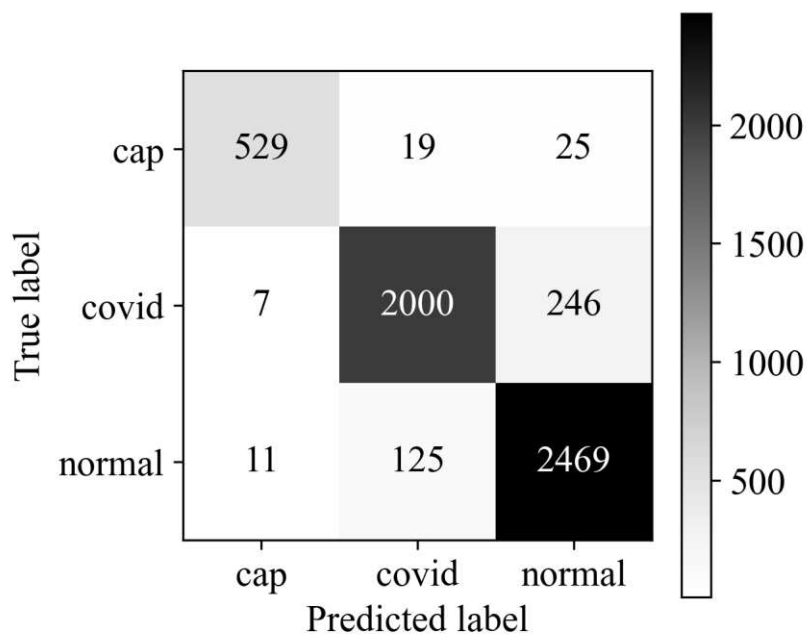
- [70] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, “Nextflow enables reproducible computational workflows,” *Nat Biotechnol*, vol. 35, no. 4, pp. 316–319, Apr. 2017, doi: 10.1038/nbt.3820.
- [71] J. Vivian *et al.*, “Toil enables reproducible, open source, big biomedical data analyses,” *Nat Biotechnol*, vol. 35, no. 4, pp. 314–316, 2017, doi: 10.1038/nbt.3772.
- [72] Šimko Tibor, Heinrich Lukas, Hirvonsalo Harri, Kousidis Dinos, and Rodríguez Diego, “REANA: A System for Reusable Research Data Analyses,” *EPJ Web Conf.*, vol. 214, p. 6034, 2019, doi: 10.1051/epjconf/201921406034.
- [73] P. Amstutz *et al.*, “Arvados,” Apr. 2024, *Zenodo*. doi: 10.5281/zenodo.10957460.
- [74] M. Rahimzadeh, A. Attar, and S. M. Sakhaei, “A fully automated deep learning-based network for detecting COVID-19 from a new and large lung CT scan dataset,” *Biomed Signal Process Control*, vol. 68, p. 102588, 2021, doi: <https://doi.org/10.1016/j.bspc.2021.102588>.
- [75] P. Afshar *et al.*, “COVID-CT-MD, COVID-19 computed tomography scan dataset applicable in machine learning and deep learning,” *Sci Data*, vol. 8, no. 1, p. 121, Apr. 2021, doi: 10.1038/s41597-021-00900-3.
- [76] W. R. Webb, W. E. Brant, and N. M. Major, *Fundamentals of Body CT*, 3rd ed. Elsevier, 2005.
- [77] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” 2017.
- [78] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
- [79] F. Heimerl and M. Gleicher, “Visual Exploration of Word Vector Embeddings,” in *IEEE Visualization Poster Proceedings*, [Online]. Available: <https://par.nsf.gov/biblio/10064420>
- [80] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” Mar. 2018.
- [81] M. Kelcey, “Metric learning for image similarity search,” *keras.io*. Accessed: May 15, 2024. [Online]. Available: https://keras.io/examples/vision/metric_learning/

- [82] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A Comprehensive Survey of Loss Functions in Machine Learning,” *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, Apr. 2022, doi: 10.1007/s40745-020-00253-5.
- [83] E. Hoffer and N. Ailon, “Deep metric learning using Triplet network,” 2018.
- [84] Y. LeCun, C. Cortes, and C. J. C. Burges, “The MNIST Database of Handwritten Digits.” Accessed: Jun. 20, 2024. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [85] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” Aug. 2017.
- [86] T. Dozat, “Incorporating Nesterov Momentum into Adam,” 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:70293087>
- [87] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” *Neurocomputing*, vol. 408, pp. 189–215, 2020, doi: <https://doi.org/10.1016/j.neucom.2019.10.118>.
- [88] P. J. and P. P. M. Mucherino Antonio and Papajorgji, “k-Nearest Neighbor Classification,” in *Data Mining in Agriculture*, New York, NY: Springer New York, 2009, pp. 83–106. doi: 10.1007/978-0-387-88615-2_4.
- [89] L. Breiman, “Random Forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [90] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011, [Online]. Available: <http://jmlr.org/papers/v12/pedregosa11a.html>
- [91] L. Rokach, “Ensemble Methods for Classifiers,” in *Data Mining and Knowledge Discovery Handbook*, L. Maimon Oded and Rokach, Ed., Boston, MA: Springer US, 2005, pp. 957–980. doi: 10.1007/0-387-25465-X_45.
- [92] R. E. Schapire, “Explaining AdaBoost,” in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, Z. and V. V. Schölkopf Bernhard and Luo, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 37–52. doi: 10.1007/978-3-642-41136-6_5.

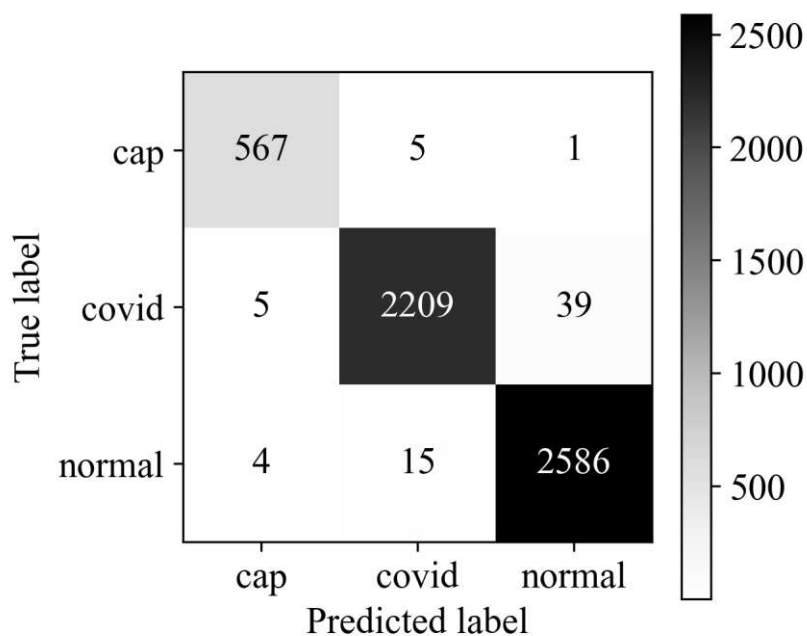
- [93] C. E. Rasmussen, “Gaussian Processes in Machine Learning,” in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, U. and R. G. Bousquet Olivier and von Luxburg, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. doi: 10.1007/978-3-540-28650-9_4.
- [94] B. Ghojogh and M. Crowley, “Linear and Quadratic Discriminant Analysis: Tutorial,” 2019. [Online]. Available: <https://arxiv.org/abs/1906.02590>
- [95] K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone, “AutoGP: Exploring the Capabilities and Limitations of Gaussian Process Models,” Oct. 2016.
- [96] C. S. Peirce, “Prolegomena to an Apology for Pragmaticism,” *Monist*, vol. 16, no. 4, pp. 492–546, 1906, doi: 10.5840/monist190616436.
- [97] L. Wetzel, *Types and Tokens: On Abstract Objects*. MIT Press, 2009. Accessed: Oct. 21, 2024. [Online]. Available: <https://books.google.com.ua/books?id=jNATqWe3B3sC>
- [98] S. Baichoo *et al.*, “Developing reproducible bioinformatics analysis workflows for heterogeneous computing environments to support African genomics,” *BMC Bioinformatics*, vol. 19, no. 1, Dec. 2018, doi: 10.1186/s12859-018-2446-1.
- [99] T. M. Porter and M. Hajibabaei, “MetaWorks: A flexible, scalable bioinformatic pipeline for high-throughput multi-marker biodiversity assessments,” *PLoS One*, vol. 17, no. 9, Sep. 2022, doi: 10.1371/journal.pone.0274260.
- [100] L. Arrabito, J. Bregeon, A. Faure, O. Gueta, N. Pigoux, and A. Tsaregorodtsev, “The Cherenkov Telescope Array Observatory workflow management system,” *EPJ Web Conf*, vol. 295, p. 04044, May 2024, doi: 10.1051/epjconf/202429504044.
- [101] P. A. Ewels *et al.*, “nf-core: Community curated bioinformatics pipelines,” *bioRxiv*, p. 610741, Jan. 2019, doi: 10.1101/610741.
- [102] J. M. Perkel, “Workflow systems turn raw data into scientific knowledge,” *Nature*, vol. 573, no. 7772, pp. 149–150, Sep. 2019, doi: 10.1038/d41586-019-02619-z.
- [103] Z. He, Y. Pan, F. Shao, and H. Wang, “Identifying Differentially Expressed Genes of Zero Inflated Single Cell RNA Sequencing Data Using Mixed Model Score Tests,” *Front Genet*, vol. 12, Feb. 2021, doi: 10.3389/fgene.2021.616686.

- [104] X. Xie, H. Liu, W. Hou, and H. Huang, “A Brief Survey of Vector Databases,” in *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, IEEE, 2023, pp. 364–371.
- [105] M. Ghayoumi, M. Gomez, K. E. Baumstein, N. Persaud, and A. J. Perlowin, “Local Sensitive Hashing (LSH) and Convolutional Neural Networks (CNNs) for Object Recognition,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, Dec. 2018, pp. 1197–1199. doi: 10.1109/ICMLA.2018.00193.
- [106] Y. A. Malkov and D. A. Yashunin, “Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs,” *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 4, pp. 824–836, Apr. 2020, doi: 10.1109/TPAMI.2018.2889473.
- [107] Q. Yang *et al.*, “Ultra-fast and accurate electron ionization mass spectrum matching for compound identification with million-scale in-silico library,” *Nat Commun*, vol. 14, no. 1, p. 3722, Jun. 2023, doi: 10.1038/s41467-023-39279-7.
- [108] “Pinecone: the vector database to build knowledgeable AI.” Accessed: Aug. 15, 2024. [Online]. Available: <https://www.pinecone.io/>
- [109] J. Wang *et al.*, “Milvus: A Purpose-Built Vector Data Management System,” in *Proceedings of the 2021 International Conference on Management of Data*, New York, NY, USA: ACM, Jun. 2021, pp. 2614–2627. doi: 10.1145/3448016.3457550.
- [110] “Weaviate - Vector Database.” Accessed: Aug. 15, 2024. [Online]. Available: <https://weaviate.io/>
- [111] N. Matsumoto *et al.*, “KRAGEN: a knowledge graph-enhanced RAG framework for biomedical problem solving using large language models,” *Bioinformatics*, vol. 40, no. 6, Jun. 2024, doi: 10.1093/bioinformatics/btae353.
- [112] N. Provos and D. Mazières, “A Future-Adaptable Password Scheme,” in *Proceedings of the FREENIX Track:1999 USENIX Annual Technical Conference*, 1999.

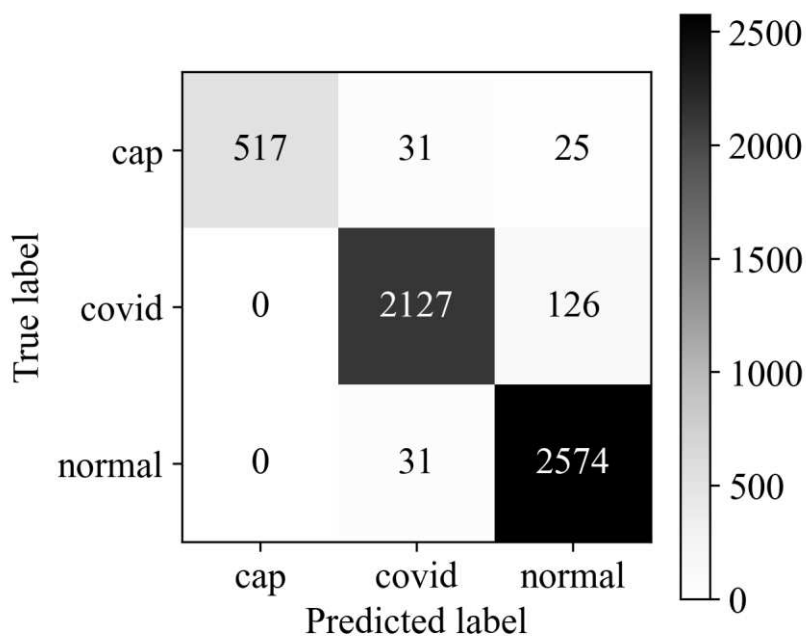
**ДОДАТОК А. МАТРИЦІ НЕВІДПОВІДНОСТІ КЛАСИФІКАТОРІВ,
НАВЧЕНИХ НА ВКЛАДЕНИХ ПРЕДСТАВЛЕННЯХ ЗНІМКІВ КТ**



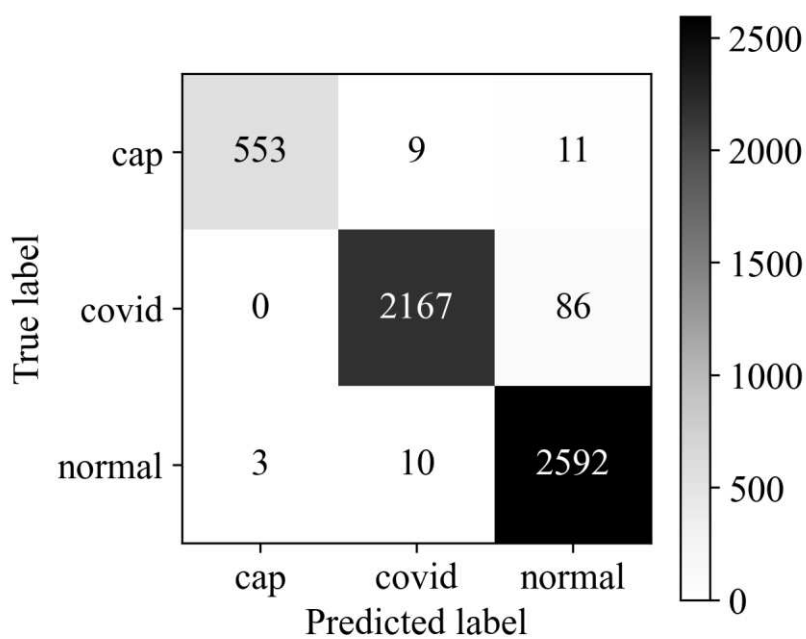
Матриця невідповідності для опорних векторів



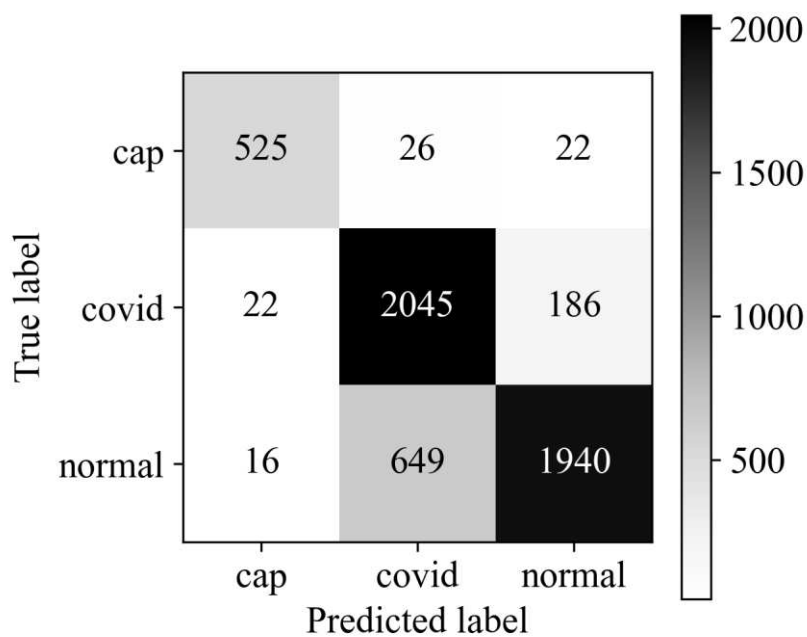
Матриця невідповідності для методу k-найближчих сусідів



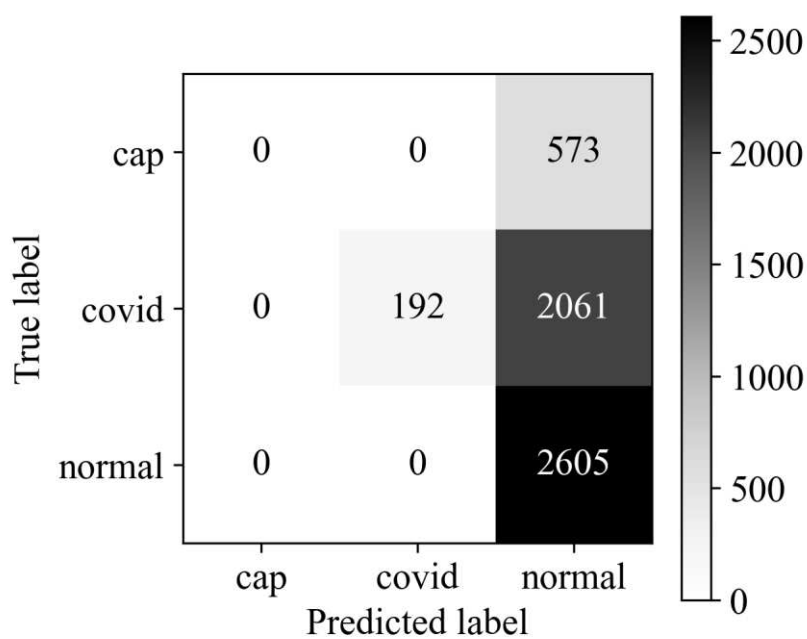
Матриця невідповідності для методу випадкового лісу



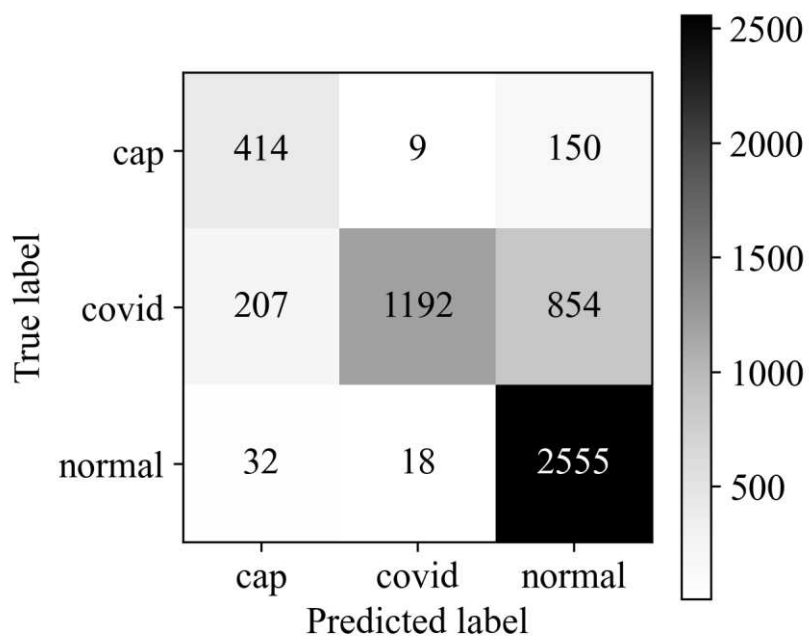
Матриця невідповідності для ансамблевого класифікатора



Матриця невідповідності для методу AdaBoost



Матриця невідповідності для методу Гаусівського процесу



Матриця невідповідності для методу аналізу квадратичних дискримінант

ДОДАТОК Б. ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОГРАМНИМ ЗАСОБОМ

Розглянемо процес виконання аналізу знімків за допомогою розробленого ПЗ.

Для цього необхідними є такі операції:

- створення адміністратором організації від імені якої здійснюватиметься аналіз (якщо ще не створено);
- внесення користувача у організацію адміністратором ПЗ чи організації;
- реєстрація користувача у ПЗ;
- створення аналізу;
- додавання вхідних даних;
- розгляд результатів.

Створення організації відбувається за допомогою панелі адміністратора (рис. Б.1).

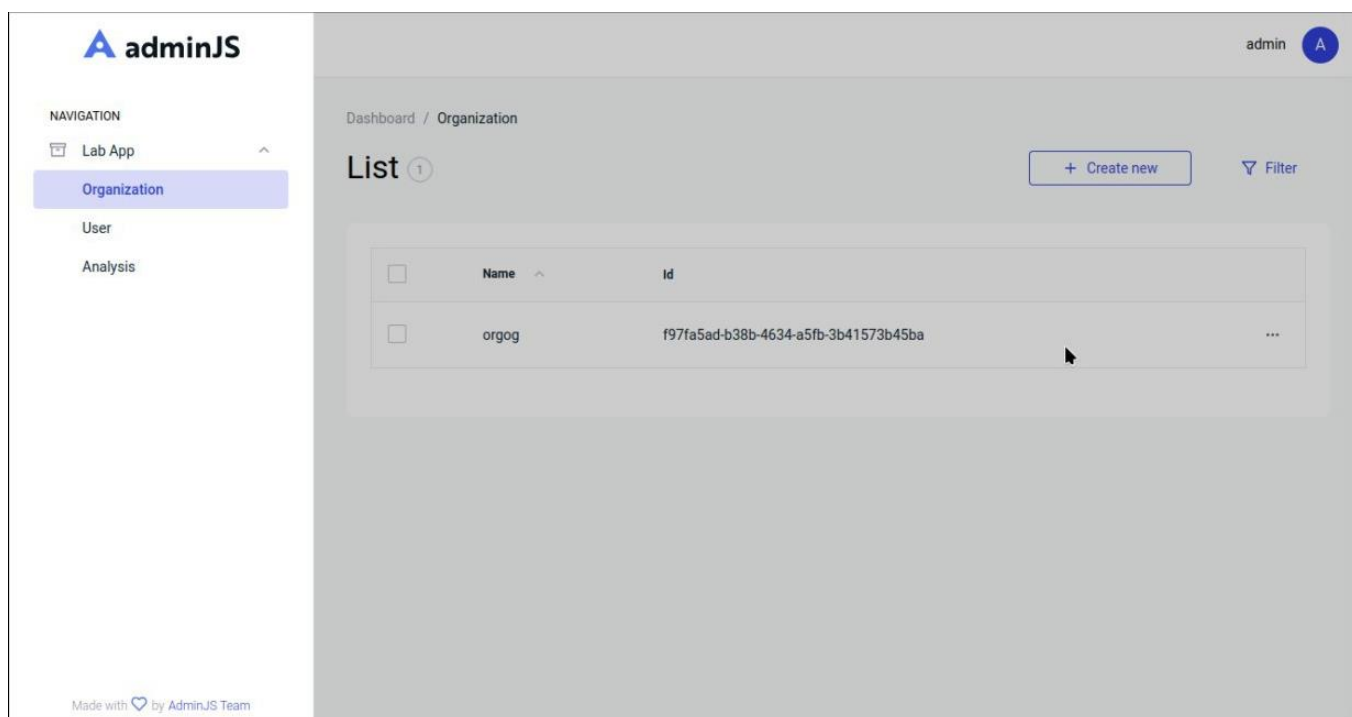
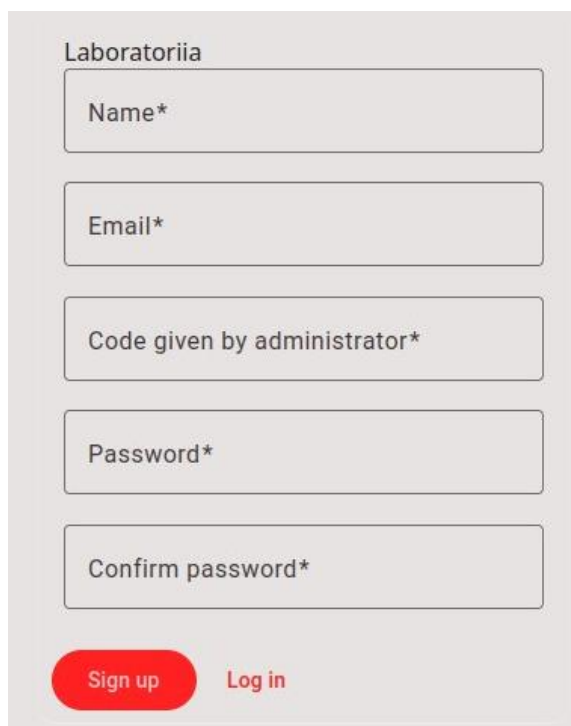


Рисунок Б.1 – Панель адміністратора ПЗ

При першому вході на платформу користувач вказує реєстраційний код (для унеможливлення «вгадування» поштової скриньки нещодавно внесених користувачів) та встановлює пароль (рис. Б.2).



Laboratoriia

Name*

Email*

Code given by administrator*

Password*

Confirm password*

Sign up Log in

Рисунок Б.2 – Перший вхід користувача у систему

Після входу в систему користувач має доступ до аналізів, що були створені в організації до якої він належить. Кожен аналіз представлений у списку із зазначенням назви, дати створення та останнього оновлення, а також результату аналізу, якщо його було завершено. За допомогою даного екрану користувач може знайти існуючий аналіз або створити новий, вказавши назву та, за потреби, примітки для додаткових даних (рис. Б.3).

Створивши чи відкривши існуючий аналіз, користувач може додати до нього дані та запустити проведення аналізу. Під час виконання аналізу користувацький інтерфейс виконує запити кожні 5 секунд для перевірки готовності результату. Результат відображається у момент готовності (рис. Б.4).

Laboratoriia

Analyses

| Name | Created At | Updated At |
|-------------|-------------|-------------|
| normal037 | Nov 8, 2024 | Nov 8, 2024 |
| normal015 | Nov 8, 2024 | Nov 8, 2024 |
| normal009 | Nov 8, 2024 | Nov 8, 2024 |
| covid133 | Nov 8, 2024 | Nov 8, 2024 |
| covid122 | Nov 8, 2024 | Nov 8, 2024 |
| covid067 | Nov 8, 2024 | Nov 8, 2024 |
| cap034 | Nov 8, 2024 | Nov 8, 2024 |
| cap029 | Nov 8, 2024 | Nov 8, 2024 |
| cap028-real | Nov 8, 2024 | Nov 8, 2024 |
| cap028 | Nov 8, 2024 | Nov 8, 2024 |

Items per page: 10 1 - 10 of 35

Рисунок Б.3 – Список створених аналізів

Laboratoriia

cap025

Data:

- cap025.zip
Kind: **dicom-chest-ct-image-zip**
[Download](#)
- tiff-out.zip
Kind: **tiff-chest-ct-image-filtered-zip**
Produced by: dicom-zip-filter-to-tiff
[Download](#)
- result.json
Kind: **chest-ct-slice-embeddings**
Produced by: dicom-zip-filter-to-tiff => resnet-fdn-embed

Steps taken:

- dicom-zip-filter-to-tiff
success in 20.20 seconds
- resnet-fdn-embed
success in 25.27 seconds
- resnet-fdn-classify
success in 25.18 seconds
- neigh-embed-classify
success in 10.09 seconds
- randfor-embed-classify
success in 15.12 seconds
- ada-embed-classify
success in 15.11 seconds

Results:

Status: complete
Result: **cap**
Total analyzed images: 32

Per method:
resnet-fdn-classify: **cap** (30 images classified as **cap**, 2 images classified as **covid**)
neigh-embed-classify: **cap** (32 images classified as **cap**)
ada-embed-classify: **cap** (32 images classified as **cap**)
randfor-embed-classify: **cap** (32 images classified as **cap**)
svc-embed-classify: **cap** (32 images classified as **cap**)

Similar:

- covid133 - covid
- cap034 - cap
- cap028-real - cap
- normal015 - normal

Рисунок Б.4 – Екран аналізу

Після завершення потоку робіт, сторінка аналізу відображає усі вхідні, проміжні та вихідні дані, обчислені під час обробки знімка, необхідні для цього кроки та деталізовану інформацію про результати класифікації зрізів знімка. Додатково, екран відображає аналізи, що містять схожі знімки, знайдені за допомогою векторної бази даних та дозволяє перейти на сторінки цих аналізів.

Розроблений програмний засіб аналізу знімків КТ для діагностики реалізує набір операцій для керування даними, персоналом та виконання діагнозів.

ДОДАТОК В. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. **F. Smilianets** and O. Finogenov, “Multi-Class Classification of Pulmonary Diseases Using Computer Tomography Images,” *Adaptive Systems of Automatic Control*, vol. 2, no. 43, pp. 78–83, Dec. 2023, doi: 10.20535/1560-8956.43.2023.292255. *Здобувачу належить реалізація мультикласового класифікатора, його експериментальне дослідження.*
2. **F. Smilianets**, “Application of Transfer Learning for enhanced pulmonary disease detection via CT image embeddings,” *Adaptive Systems of Automatic Control*, vol. 1, no. 44, pp. 24–29, 2024, doi: 10.20535/1560-8956.44.2024.302198. *Здобувачу належить реалізація генератора вкладених представлень зображень КТ, мультикласового класифікатора, його експериментальне дослідження.*
3. **F. Smilianets** and O. Finogenov, “Running a workflow without workflows: a basic algorithm for dynamically constructing and traversing an implied directed acyclic graph in a non-deterministic environment,” *Informatyka, Automatyka, Pomiar w Gospodarce i Ochronie Środowiska*, vol. 14, no. 1, pp. 115–118, Mar. 2024, doi: 10.35784/iapgos.5858 (**проіндексовано в Scopus**). *Здобувачу належить реалізація генератора вкладених представлень зображень КТ, мультикласового класифікатора, його експериментальне дослідження.*
4. **F. Smilianets**, “Application of embeddings for multi-class classification with optional extendability,” *Adaptive Systems of Automatic Control*, vol. 2, no. 45, pp. 186–193, Oct. 2024, doi: 10.20535/1560-8956.45.2024.313198. *Здобувачу належить реалізація реалізація мультикласових класифікаторів, дослідження втрати точності при додаванні нового класу.*
5. **F. A. Smilianets** and O. D. Finogenov, “Review of disease identification methods based on computed tomography imagery,” *Ukrainian Journal of Information Technology*, vol. 6, no. 1, pp. 95–100, 2024, doi: 10.23939/ujit2024.01.095. *Здобувачу належить збір та аналіз інформації про існуючі дослідження.*

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. **Смілянець Ф.А.**, Фіногенов О.Д. Огляд методів ідентифікації захворювань на основі знімків комп'ютерної томографії. XII Міжнародна науково-практична конференція «Комплексне забезпечення якості технологічних процесів та систем», 2022, ISBN 978-617-7932-34-4. *Здобувачу належить збір та аналіз інформації про існуючі дослідження.*
2. **Смілянець, Ф. А.**, Фіногенов О.Д. Мультикласова класифікація легеневих захворювань за допомогою знімків комп'ютерної томографії. Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023): матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів, присвячених 125-й річниці КПІ ім. Ігоря Сікорського (9-11 травня 2023 р., Київ). – Київ : КПІ ім. Ігоря Сікорського, ІІІ ФІОТ, 2023. с. 28-30. *Здобувачу належить реалізація мультикласового класифікатора, його експериментальне дослідження.*
3. **Смілянець, Ф. А.**, Фіногенов О.Д. Методи та програмне забезпечення обробки знімків комп'ютерної томографії для автоматизації діагностики захворювань. Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2024): матеріали VI Міжнародної науково-практичної конференції молодих вчених та студентів, 21-23 травня 2024 року – Київ : КПІ ім. Ігоря Сікорського, ІІІ ФІОТ, 2024. с. 124. *Здобувачу належить розробка та обґрунтування запропонованих методів.*
4. **Смілянець, Ф. А.**, Фіногенов О.Д. Порівняння алгоритмів класифікації з використанням вкладених представлень знімків КТ. Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2024): матеріали VII Міжнародної науково-практичної конференції молодих вчених та студентів, 20-22 листопада 2024 року – Київ : КПІ ім. Ігоря Сікорського, ІІІ ФІОТ, 2024. с. 96-100. *Здобувачу належить проведення обчислювальних експериментів та порівняння алгоритмів класифікації з використанням вкладених представлень.*