НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

Завідувач кафедри

_____Олена ЧУМАЧЕНКО

«___» _____ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи і методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки»

на тему: «Біометрична автентифікація користувача смартфона за допомогою даних акселерометра»

Виконав :
студент IV курсу, групи КІ-93
Кагарлицький Роман Євгенійович _____

Керівник:
професор, д.т.н., Данилов Валерій Якович _____

Консультант з нормконтролю:
Гончарук Максим Миколайович _____

Консультант з економічного розділу:
доцент, к.е.н., Рощина Надія Василівна _____

Рецензент:
професор, д.т.н., Новіков Олексій Миколайович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ, 2023

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітня програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ:

Завідувач кафедри

_____Олена ЧУМАЧЕНКО

«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Кагарлицькому Роману Євгенійовичу

1. Тема роботи «Біометрична автентифікація користувача смартфона за допомогою даних акселерометра», керівник роботи професор, д.т.н. Данилов Валерій Якович, затверджені наказом по університету від «30» травня 2023 р. № 2065-с
2. Термін подання студентом роботи 08.06.2023
3. Вихідні дані до роботи – набір акселерометричних даних, зібраних різних користувачів за допомогою вбудованого у смарфтон сенсора.
4. Зміст роботи – 1. Дослідження предметної області; 2. – Математичні основи робота; 3. Аналіз результатів роботи; 4. Функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доцент, к.е.н., Рощина Н.В.		

7. Дата видачі завдання 21.02.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Підготовка даних до роботи	27.02.2023	Виконано
2.	Вивчення літератури за темою роботи	25.03.2023	Виконано
3.	Підготовка першого розділу	10.04.2023	Виконано
4.	Розробка нейронних мереж	16.04.2023	Виконано
5.	Підготовка другого розділу	08.05.2023	Виконано
6.	Розробка веб-клієнта	20.05.2023	Виконано
7.	Підготовка третього розділу	24.05.2023	Виконано
8.	Підготовка економічної частини	01.06.2023	Виконано
9.	Підготовка презентації доповіді	03.06.2023	Виконано
10.	Оформлення дипломної роботи	05.06.2023	Виконано

Студент Роман КАГАРЛИЦЬКИЙ

Керівник Валерій ДАНИЛОВ

АНОТАЦІЯ

Дипломна робота: 122 сторінки, 24 рисунки, 14 таблиць, 1 додаток, 39 джерел.

МАШИННЕ НАВЧАННЯ, КОНВОЛЮЦІЙНІ НЕЙРОННІ МЕРЕЖІ, АКСЕЛЕРОМЕТЕР, ЧАСОВІ РЯДИ, СКАЛЕОГРАМА, ВІКОННЕ ПЕРЕТВОРЕННЯ ФУР'Є, ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ.

Об’єкт дослідження – розробка нейронної мережі для задачі автентифікації користувача смартфона за допомогою даних акселерометра.

В наші дні майже у кожної людини є смартфон. Ми використовуємо їх для спілкування, ведення соціальних мереж, для банківських операцій тощо. Для захисту цих пристроїв ми налаштовуємо коди, паролі, шаблони свайпів та біометричні методи, такі як розпізнавання відбитків пальців та обличчя. Це приклади методів одноразової автентифікації; вони ґрунтуються на тому, що після розблокування телефону законним користувачем він буде єдиним, хто зможе ним користуватися, доки телефон не буде повторно заблокований.

Традиційні методи автентифікації (перелічені вище) вимагають від користувача виконання певних дій для логування в систему. Це негативно впливає на користувацький досвід, адже змушує користувача виконувати рутинну роботу.

Мета цієї роботи полягає в створенні оптимального методу біометричної автентифікації користувача смартфона з використанням даних акселерометра, застосовуючи алгоритми і підходи глибинного навчання.

Розробка такого додаткового пасивного рівня захисту зробило б наші смартфони простішими в користуванні, більш захищеними, і при цьому не вимагало б якихось складних постійних обчислень – адже отримати й обробити дані вбудованого в смартфон акселерометра дуже просто в наші дні і це не потребує потужного процесора.

ABSTRACT

Bachelor thesis: 122 pages, 24 figures, 14 tables, 1 appendix, 39 sources.

MACHINE LEARNING, CONVOLUTIONAL NEURAL NETWORKS, ACCELEROMETER, TIME SERIES, SCALEOGRAM, SHORT-TIME FOURIER TRANSFORM, WAVELET TRANSFORM.

Research object is the development of a neural network for the task of authenticating a smartphone user using accelerometer data.

These days, almost everyone has a smartphone. We use them for communication, social networking, banking, etc. To protect these devices, we set up codes, passwords, swipe patterns, and biometric methods such as fingerprint and face recognition. These are examples of one-time authentication methods; they are based on the idea that once a phone is unlocked by a legitimate user, they are the only one who can use it until the phone is re-locked.

Traditional authentication methods (listed above) require the user to perform certain actions to log in to the system. This has a negative impact on the user experience, as it forces the user to perform routine work.

The purpose of this work is to create an optimal method of biometric authentication of a smartphone user using accelerometer data, applying machine learning algorithms and deep learning approaches.

The development of such a additional passive layer of protection would make our smartphones easier to use, more secure, and would not require any complex constant computing - after all, it is very easy to obtain and process data from the accelerometer built into a smartphone these days and it does not require a powerful processor.

ЗМІСТ

ВСТУП

У 21 столітті майже у кожної людини є смартфон. Все нові й нові застосунки з’являються щодня, розширюючи і так доволі багатий функціонал цих пристроїв. Смартфони використовуються для спілкування, ведення соціальних мереж, для банківських операцій, доступу до інтернету, зберігання приватної інформації і фотографій. Для захисту цих пристроїв ми налаштовуємо коди, паролі, шаблони свайпів та біометричні методи, такі як розпізнавання відбитків пальців та обличчя [1].

Але, існуючі системи ідентифікації користувачів не є досконалими. І всі вони стикаються з різними соціальними та практичними проблемами. Наприклад, ідентифікація за відбитками пальців вимагає контакту пальця з датчиком, і навіть незначна деградація поверхні або забруднення можуть спричинити проблему. Розпізнавання обличчя або райдужної оболонки ока вимагає достатнього рівня освітлення, а розпізнавання голосу - відсутності сторонніх звуків. З іншого боку, методи, що базуються на паролях, вимагають великого запам'ятовування і можуть бути вкрадені або загублені.

Це приклади методів одноразової автентифікації; вони ґрунтуються на тому, що після розблокування телефону законним користувачем він буде єдиним, хто зможе ним користуватися, доки телефон не буде повторно заблокований. Такі додатки, як App Store від Apple та більшість банківських додатків, вимагають повторної автентифікації для здійснення покупок або отримання доступу.

Отже, традиційні методи автентифікації (перелічені вище) вимагають від користувача виконання певних дій для логування в систему. Це негативно впливає на користувацький досвід, адже змушує користувача виконувати рутинну роботу. Понад 31% користувачів смартфонів не використовують паролі або PIN-коди на своїх пристроях через поєднання легковажності та поганого користувацького досвіду [2].

Як щодо того, щоб для задачі автентифікації використовувати дані акселерометру? Це можливо?

Так, можливо. Все це завдяки унікальним характеристикам руху, які можуть ідентифікувати конкретну людину. Акселерометр є датчиком, який вимірює прискорення, яке діє на смартфон у трьох основних напрямках: вперед-назад, вліво-вправо та вгору-вниз [3]. І, оскільки стиль руху кожної людини є унікальним, дані акселерометра можуть бути використані для ідентифікації користувача. Наприклад, спосіб, яким користувач тримає смартфон, його швидкість руху, ритм кроку або навіть мікрохаотичність рухів можуть бути унікальними для кожної особи.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність роботи

Розвиток сучасних технологій безпеки та зручності використання мобільних пристроїв робить актуальною роботу в галузі біометричної автентифікації користувача смартфона за допомогою даних акселерометра. Захист особистих даних користувачів стає все більш критичним завданням, оскільки смартфони використовуються для зберігання конфіденційної інформації, фінансових даних та іншого.

Штучний інтелект здатний автоматично вивчати та розпізнавати унікальні характеристики кожного користувача, пов'язані з рухом та орієнтацією смартфона. Це дозволяє створити персоналізовані моделі, які забезпечують високу точність та надійність автентифікації.

Дослідження в цій області може мати значний практичний вплив на забезпечення безпеки особистих даних та зручність використання смартфона. Використання даних акселерометра для автентифікації дозволить збільшити рівень захисту від несанкціонованого доступу, зменшити необхідність в додаткових апаратних засобах та сприяти швидкому та зручному використанню смартфона користувачем.

Процес автентифікації на основі акселерометра включатиме збір і аналіз даних про рух користувача в реальному часі або порівняння цих даних з попередньо збереженими шаблонами руху. Зазвичай використовуються алгоритми машинного навчання або штучні нейронні мережі для виявлення унікальних ознак руху користувача та порівняння їх зі збереженими шаблонами.

Однією з переваг автентифікації на основі акселерометра є те, що вона може бути реалізована без потреби в додатковому апаратному забезпеченні або складних сенсорах. Акселерометр є стандартним компонентом багатьох

смартфонів і мобільних пристроїв, тому він доступний для використання без додаткових витрат.

У численних дослідженнях [17, 19, 21] методи безперервної пасивної автентифікації для мобільних телефонів показали свою перспективність.

Проте, варто зазначити, що автентифікація на основі акселерометра має свої обмеження і не є 100% надійним методом. Чинники, такі як зміна умов освітлення, енергійна ефективність пристрою або можливість імітації руху, можуть впливати на точність і надійність автентифікації. Тому, для підвищення безпеки, рекомендується комбінувати автентифікацію на основі акселерометра з іншими методами, такими як сканування відбитку пальця або розпізнавання обличчя.

Отже, таке програмне рішення повинне бути здатним ефективно, безшумно і ненав'язливо розпізнавати реального користувача за даними датчиків на смартфоні і таким чином автоматично забезпечувати необхідний захист конфіденційності та автентифікацію. У зв'язку з цим, у цій роботі представлено систему ідентифікації користувачів на основі ходи, яка використовує показники датчиків, вбудованих у смартфон.

Хода пов'язана з особливостями та манерою ходьби людини, і вона досить сильно відрізняється у кожної людини. Ідентифікація за ходою є одним з найновіших напрямків біометрії, який набуває потенціалу для вирішення низки проблем. У підході, заснованому на ході, немає необхідності в безпосередньому контакті з суб'єктом. Це дослідження фокусується виключно на вбудованих у смартфони датчиках MEMS (мікроелектромеханічних системах), які записують дані акселерометра людини під час ходьби. Сигнали акселерометра аналізуються для визначення характерної ходи людини, що дозволяє розпізнавати дані від багатьох постачальників. Тут задача ідентифікації користувача перетворюється на задачу класифікації зображень шляхом перетворення сигналу акселерометра у 2D-частотно-часову репрезентацію зображення за допомогою обчислення вейвлет-аналізу сигналу. Зокрема, акселерометр у смартфонах вимірює і записує

дані за трьома осями X, Y і Z. Отже, дані, що надаються акселерометром, мають характер часових рядів, і таких рядів може бути три. Теоретично, ці дані повинні містити закономірність, коли людина носить смартфон під час ходьби, і ця закономірність повинна бути унікальною для кожної людини [3]. А, якщо патерн можна ідентифікувати, то можна виконати завдання ідентифікації користувача.

Для цього дані часового ряду для кожної людини сегментуються на невеликі вікна, і кожне вікно перетворюється на частотно-часову репрезентацію за допомогою обчислення вейвлет-аналізу сигналу. Ця репрезентація тепер подається на модель глибокого навчання на базі CNN (конволюційної нейронної мережі) [4], яка класифікує її відповідним чином.

Моя мотивація для цього дослідження в основному походить від обмежень існуючих рішень. Зокрема, мотивація полягає в тому, щоб вирішити наступні два недоліки існуючих робіт:

1. Відсутність ефективних методів вилучення ознак: у більшості існуючих робіт (див. Таблицю 1.1) використовували підходи, засновані на поверхневому машинному навчанні або глибокому навчанні, для розпізнавання користувачів за даними сигналів акселерометра. Вони зосереджені більше на прогностичних моделях (predictive models), ніж на ефективних методах вилучення ознак.
2. Відсутність ефективних легких моделей для пасивної автентифікації на базі смартфонів: незважаючи на те, що сьогодні доступно багато великих і просунутих моделей глибокого навчання, їх неможливо використовувати пасивно в смартфонах для розпізнавання користувачів, адже обслуговування (deployment, serving) дуже великих моделей у системах на базі смартфонів призводить до значного обчислювального навантаження та навантаження на пам'ять смартфона, а також до значного виснаження заряду акумулятора [15]. Отже, для системи розпізнавання користувачів на базі смартфона потрібна легка, але ефективна модель.

Model	Reference	Approach/description	Accuracy	Precision	Recall	#Samples	#Sensors	#Users
Randomized CNN	Oguz et al. [17]	User identification based on accelerometer data using basic statistical feature extraction like entropy, skewness, kurtosis, standard deviation	97	–	–	3000	1	387
RF	Johnston and Weiss [18]	Random forest model for gait-based biometric identification using accelerometer and gyroscope data	84%	–	–	–	2	59
Conv-LSTM	Sun et al. [19]	Using 1D-convolutional long short-term memory neural networks, this work implements an EEG-based user identification system	98	98	98	10,240	1	109
LLR	Cola et al. [20]	User identification using Multinomial Logistic Regression (LLR)	99	–	95	1536	1	10
DT	Liu et al. [21]	User authentication in wireless body area network based on accelerometer data using C4.5 decision tree classifier	86.70%	–	–	–	1	7
CNN	Wang et al. [22]	Feature Integration and weighted sub carrier screening are used for better performance	98	–	–	755	2	15
NN	Lee et al. [23]	User identification	95	–	–	1386	9	14

Model	Reference	Approach/description	Accuracy	Precision	Recall	#Samples	#Sensors	#Users
		using signal collected from pressure sensors and accelerometer present in insole of shoes						
RF	Weiss et al. [24]	User identification or authentication based on smartphone and smartwatch accelerometer sensor data using random forest algorithm	94.70%	–	–	–	2	51
DNN	Volaka et al. [25]	Continuous of user authentication on mobile phones using deep neural networks (DNN)	88%	78-79%	–	–	2	100
CNN	Benegui et al. [26]	User identification using the image representation of smartphone equipped accelerometer and the gyroscope data	90.75	–	–	150	2	50
LSTM	Watanabe et al. [27]	LSTM-based user identification and authentication based on smartphones' 3-axis accelerometer data	95%	–	–	–	1	21
RF	Angrisano et al. [28]	Walker identity recognition based on MEMS sensors' data using ensemble learning	93.8	92.0	93.0	32	3	18
CNN	Neverova et al. [29]	Compare various deep learning models (Conv-DCWRNN, Conv-CWRNN, Conv-LSTM, etc.) for user identity	69.41	–	–	500	3	587

Model	Reference	Approach/description	Accuracy	Precision	Recall	#Samples	#Sensors	#Users
		recognition task						
SVM	Vhaduri et al. [30]	Using coarse-grained samples, identify wearable users in a continuous manner	93%	–	–	–	5	500
RF	Ehatisham-ul-haq et al.[31]	Uses wearable sensors' data to recognize users based on activity patterns	85%	–	–	–	4	9
NN	Buriro et al. [32]	Identification of users based on micro and macro hand movements on the phone screen	–	–	71.0	175	3	30
NB	Shi et al. [33]	Passive user identity recognition using smartphone sensors' data by a novel framework called Senguard	96.4	–	–	512	1	4
Heuristic	Damaeviius et al. [34]	For user identity recognition, heuristic gait features and random projection technique are used for feature dimensionality reduction	95.52%	–	–	–	2	14
KNN	Kumar et al. [35]	KNN-based approach for user authentication using the users' arm movements data while walking	95%	–	–	–	1	12
NN	Kwapisz et al. [36]	Walker identity recognition based on neural network (NN) model from accelerometer signal	74.7	–	–	200	1	36

Model	Reference	Approach/description	Accuracy	Precision	Recall	#Samples	#Sensors	#Users
J48	Kwapisz et al. [36]	Walker identity recognition based on J48 classification model from accelerometer signal	79.3	–	–	200	1	36
KNN	Yam et al. [37]	Identity recognition by extracting gait signatures from image analysis	85.0	–	–	–	1	20
SVM	Thang et al. [38]	Accelerometer data based gait identification using SVM	93%	–	–	–	1	11

Таблиця 1.1

Для того, щоб подолати вищезгадані обмеження, в цьому дослідженні я запропонував легку нейронну мережу на основі конволюцій (CNN), яка забезпечує хорошу точність з більшим акцентом на ефективну техніку вилучення ознак, таку як Вейвлет-перетворення. У порівнянні з існуючими передовими моделями глибокого навчання, запропонована модель CNN є легкою, ефективною і простою, з лише 4,071,598 параметрами. Варто зазначити, що ця кількість параметрів набагато менша, ніж у типових важких моделей на основі глибокого навчання, таких як VGG-16 [16]. VGG-16, наприклад, включає 138 мільйонів параметрів, що робить її надзвичайно трудомісткою, а отже, не може бути доцільною для використання в системі розпізнавання користувачів на базі смартфонів.

Набір даних, використаний у цьому дослідженні, базується на даних біометричних змагань з акселерометрії Kaggle [5]. Цей набір даних містить понад шістдесят мільйонів зразків даних акселерометра від 387 суб'єктів.

Всього було створено по 492 фотографії (scaleogram) для кожного користувача, тобто в сумі мій навчальний датасет містить 24600 зображень і 50 класів.

1.2 Опис предметної області

1.2.1 Методи одноразової автентифікації

Методи одноразової автентифікації забезпечують обмежений рівень безпеки при доступі до смартфону. Ці методи вимагають, щоб користувач ідентифікував себе один раз при початковому використанні пристрою. Проте, після розблокування смартфон залишається розблокованим до тих пір, поки не стане неактивним протягом певного періоду часу або не буде заблокований вручну.

Багато смартфонів дозволяють користувачеві налаштувати автоматичний час блокування пристрою після його неактивності. Наприклад, користувач може встановити, щоб смартфон автоматично заблоковувався через 1 хвилину, 5 хвилин, 15 хвилин або інший вибраний період (залежить від моделі телефону). Це допомагає зменшити ризик несанкціонованого доступу до пристрою, якщо він залишається без нагляду.

Однак, існує також можливість ручного блокування смартфона користувачем. Це означає, що користувач може активувати функцію блокування пристрою, коли він не використовується, навіть якщо встановлений час автоматичного блокування ще не минув. Це дає користувачу більшу гнучкість і контроль над безпекою свого смартфона.

Недоліком методів одноразової автентифікації є те, що розблокований смартфон може бути доступним для крадіжки протягом певного часу, поки не заблокується або не буде заблокований вручну. Це створює можливість для зловмисників отримати незаконний доступ до особистої інформації, даних, програм або сервісів, що може викликати серйозні наслідки для користувача [6].

1.2.2 Безперервні методи автентифікації

Для боротьби з проблемою легкого доступу до розблокованих смартфонів, деякі дослідники [20, 21] пропонують використання безперервних методів автентифікації. Ці методи включають як біометричні, так і небіометричні підходи. Основна ідея безперервної автентифікації полягає в повторній автентифікації користувача пристрою через певний проміжок часу. Враховуючи те, що середньостатистичний користувач витрачає на своєму смартфоні лише близько 65 секунд за один сеанс [8], інтервал для повторної автентифікації повинен бути достатньо коротким, щоб не заважати користувачу.

Методи безперервної автентифікації показали себе як багатообіцяючі рішення, проте наразі вони ще не широко використовуються в смартфонах відомих виробників. Великі компанії, що розробляють смартфони, можуть зосередитись на інших аспектах безпеки або необхідності розширення функціональності пристроїв. Також можуть виникати виклики в області енергоефективності та впливу безперервної автентифікації на тривалість роботи батареї смартфона.

Однак, деякі дослідники продовжують працювати над розвитком безперервних методів автентифікації з метою підвищення рівня безпеки смартфонів. Вони досліджують різні підходи, такі як поєднання біометричних і небіометричних методів, використання контекстуальної інформації (наприклад, місцезнаходження, активність користувача), а також впровадження машинного навчання і штучного інтелекту для поліпшення точності автентифікації.

Можливе майбутнє застосування безперервних методів автентифікації включає їх використання не лише для розблокування смартфонів, але й для безперервного моніторингу і ідентифікації користувача протягом усього часу використання пристрою. Це може допомогти попереджати несанкціонований доступ до особистої інформації і додатково забезпечити безпеку даних на смартфоні.

1.2.3 Пасивні методи автентифікації

Пасивні методи автентифікації націлені на те, щоб автентифікувати користувача без явного введення даних. Вони використовують інформацію, яка зберігається на смартфоні. Пасивна автентифікація може включати використання датчиків руху [20] або шаблонів використання, таких як частота дотиків та свайпів [21]. Ці методи поліпшують взаємодію з користувачем, зменшуючи затримку, яку вводять інші методи автентифікації, такі як ПІН-коди, шаблони обличчя або відбитки пальців.

Використання датчиків руху дозволяє смартфону автоматично розпізнавати певні рухи або жести користувача, що можуть слугувати як ідентифікатор. Наприклад, смартфон може розпізнавати унікальний шаблон ходьби користувача або спосіб тримання пристрою, кут нахилу й різкість рухів, тощо. Такі рухові шаблони можуть бути використані для безпечного розблокування смартфона.

Щодо шаблонів використання, смартфон може відстежувати і аналізувати частоту і стиль дотиків або свайпів, які здійснює користувач. Це може бути особистим ідентифікатором, який допомагає встановити, чи є поточний користувач справжнім власником смартфона.

Один з головних переваг пасивних методів автентифікації полягає в тому, що вони не вимагають введення додаткових даних або дій від користувача. Це робить їх зручними і швидкими у використанні, відкидаючи (частково) необхідність запам'ятовування ПІН-кодів або інших паролів. Однак, варто пам'ятати, що пасивні методи автентифікації можуть бути менш надійними порівняно з біометричними методами, такими як сканування відбитків пальців або розпізнавання обличчя, оскільки їх можна легше підробити або обманути.

Тому найкращим рішенням було б комбінувати обидва підходи. Це було б зручніше, ефективніше та безпечніше.

1.2.4 Автентифікація на основі знань

Одним із найпоширеніших методів автентифікації, які базуються на знаннях користувача, є використання PIN-кодів та графічних шаблонів. Ці методи передбачають, що користувач обирає та пам'ятає свій особистий PIN-код, пароль або графічний шаблон, який потрібно ввести для автентифікації на пристрої.

Якщо користувач обирає складний та довгий PIN-код або пароль, це може забезпечити високий рівень захисту його пристрою. Однак, на жаль, багато людей надають перевагу коротким та простим комбінаціям, оскільки це полегшує їм користування пристроєм і запам'ятовування введеного коду [9].

Дослідження також показали, що багато користувачів використовують один і той же пароль на різних платформах та сервісах. Це створює потенційну загрозу, оскільки якщо такий пароль стає відомим зловмисникам, вони можуть нанести шкоду користувачу не лише на одній платформі, а й на всіх інших, де використовується та ж сама комбінація.

Отже, важливо обрати надійний та унікальний пароль для кожної платформи та пристрою, щоб зменшити ризик несанкціонованого доступу до особистої інформації. Також рекомендується використовувати двофакторну аутентифікацію та інші додаткові заходи безпеки для забезпечення надійного захисту особистих даних.

1.2.5 Біометрична автентифікація

Біометричні методи автентифікації можна розділити на дві основні категорії: біометрично-фізіологічні та біометрично-контекстні. Біометрично-фізіологічна категорія складається з методів, які покладаються на унікальність фізіологічних характеристик. До них відносяться відбитки пальців, риси

обличчя, сканування райдужної оболонки ока, форма вух тощо. Видатні технологічні компанії, такі як Apple та Samsung, впровадили біометрично-фізіологічні методи у свої пристрої. Наприклад, Touch ID та Face ID від Apple використовують відбитки пальців та риси обличчя для автентифікації відповідно. Apple стверджує, що ймовірність помилкового розблокування iPhone чужим відбитком пальця становить лише 1 до 50 000 [10].

З іншого боку, біометрично-контекстні методи покладаються на унікальність характеристик користувача. Ці характеристики можуть включати голос користувача, ходу, патерни рухів, стиль почерку та інші поведінкові риси. Аналізуючи та порівнюючи ці контекстні біометричні фактори, пристрій може ідентифікувати особу користувача. На відміну від фізіологічної біометрії, яка є вродженою і її важко змінити, контекстна біометрія зосереджується на поведінкових аспектах, які можуть змінюватися з часом. Ця категорія методів автентифікації додає додатковий рівень безпеки, враховуючи поведінкові патерни користувача.

Перевага біометричних методів автентифікації, як фізіологічних, так і контекстних, полягає в їхній унікальності та складності для копіювання. Фізіологічні та контекстуальні характеристики кожної людини є відмінними, що ускладнює для неавторизованих осіб завдання видати себе за неї або отримати доступ до пристрою чи системи. Однак важливо зазначити, що методи біометричної автентифікації не є надійними і можуть мати обмеження. На їхню точність і надійність можуть впливати такі фактори, як умови навколишнього середовища, зміни в зовнішності або поведінці, а також технічні вразливості.

1.2.6 Human Activity Recognition (HAR)

Суміжною задачою до моєї є розпізнавання людської активності (HAR). HAR є методом, який дозволяє ідентифікувати дії, які виконує певна особа у конкретний момент часу. Сучасні смартфони забезпечують людям можливість

взаємодіяти зі своїми пристроями майже безперервно. Це відкриває широкі можливості для відстеження дій користувачів на основі лише їхніх смартфонів.

HAR технологія була розроблена для використання на різних носимих пристроях [27] і смартфонах. Наприклад, сучасні смарт-годинники та смартфони можуть вимірювати кількість сходинок, які були подолані протягом дня. Вони також можуть визначати, чи йдете ви пішки чи біжите, чи користуєтеся велосипедом або плаваєте, а також багато іншого. HAR-системи використовують різні сенсори та функціонал. Зазвичай, вони включають в себе дані про місцезнаходження, дані акселерометра, фізіологічні сигнали та навколишнє середовище.

Однак, більшість HAR-систем можуть працювати, навіть якщо деякі з цих функцій відсутні. Наприклад, смартфони успішно використовують HAR-технологію, незважаючи на те, що вони не реєструють фізіологічні особливості користувача.

Ці методи розпізнавання активності відкривають широкі перспективи в багатьох областях, включаючи здоров'я, фітнес, аналітику спорту, контроль активності та багато іншого. Шляхом аналізу та інтерпретації даних про активності користувачів,

HAR-системи можуть забезпечити корисну інформацію та допомогти в покращенні якості життя. За допомогою цих технологій, ми можемо отримати більше уявлення про наші фізичні зусилля, стиль життя та покращити загальний рівень здоров'я.

1.2.7 Акселерометер

Акселерометр - це пристрій, який вимірює прискорення, що діє на нього. У смартфонах акселерометр використовується для визначення орієнтації пристрою, виявлення руху і вимірювання прискорення [1].

Акселерометр має тривісну (3D) структуру, тобто він вимірює прискорення у трьох взаємно перпендикулярних напрямках: x, y і z. Зазвичай, ось x спрямована горизонтально вздовж ширини пристрою, ось y спрямована горизонтально вздовж висоти пристрою, а ось z спрямована вертикально вздовж товщини пристрою.

Значення, які вимірюються акселерометром, виражаються в одиницях прискорення, зазвичай у метрах за секунду в квадраті (м/с²) або в гравітаційних одиницях (g), де 1 g відповідає приблизно 9,8 м/с².

Акселерометри присутні в багатьох смартфонах. Вони є одним із сенсорів, які можуть бути вбудовані у смартфон, але наявність акселерометра залежить від моделі і виробника пристрою. Наявність акселерометра впливає на можливість вимірювання прискорення і використання його даних для різних додатків та функцій, таких як визначення орієнтації, рухового вводу, швидкісного вимірювання тощо.

Хода кожної людини може бути різною, і це впливає на показники акселерометра. Коли людина рухається, акселерометр реєструє зміни прискорення у відповідних напрямках. Зміни прискорення можуть бути обумовлені різними факторами, такими як швидкість руху, тип руху (ходьба, біг, сходження по сходах тощо), повернення тіла і т.д. Таким чином, показники акселерометра можуть варіюватися від людини до людини в залежності від їхньої ходи та рухових активностей.

1.3 Опис та аналіз даних

Джерелом набору даних, використаного у цій роботі, є відкритий набір даних акселерометра як біометричного ідентифікатора користувачів мобільних пристроїв на сайті Kaggle [5].

Компанія Seal зібрала дані акселерометра від кількох сотень користувачів протягом декількох місяців під час звичайного використання пристроїв. Для збору даних було розміщено додаток на Android PlayStore, який в фоновому режимі відбирав дані акселерометра і передавав їх до центральної бази даних для аналізу.

Вони завантажили приблизно 60 мільйонів унікальних зразків даних акселерометра, зібраних з 387 різних пристроїв. Ці дані розділені на рівні набори для навчання і тестування. Зразки в наборі для навчання мають позначку з унікального пристрою, з якого були зібрані дані (ідентифікатор користувача). Набір для тестування розмежовується на 90 тисяч послідовних зразків з одного пристрою.

Загалом, фреймворк датчика використовує стандартну 3-осьову систему координат для зчитування значень даних. Для більшості датчиків система координат визначається відносно екрана пристрою, коли пристрій утримується в стандартній орієнтації (див. рис. 1.1). Коли пристрій утримується в стандартній орієнтації, вісь X є горизонтальною і спрямована праворуч, вісь Y є вертикальною і спрямована вгору, а вісь Z спрямована до зовнішньої сторони екрана. У цій системі координати за екраном мають від'ємні значення Z.

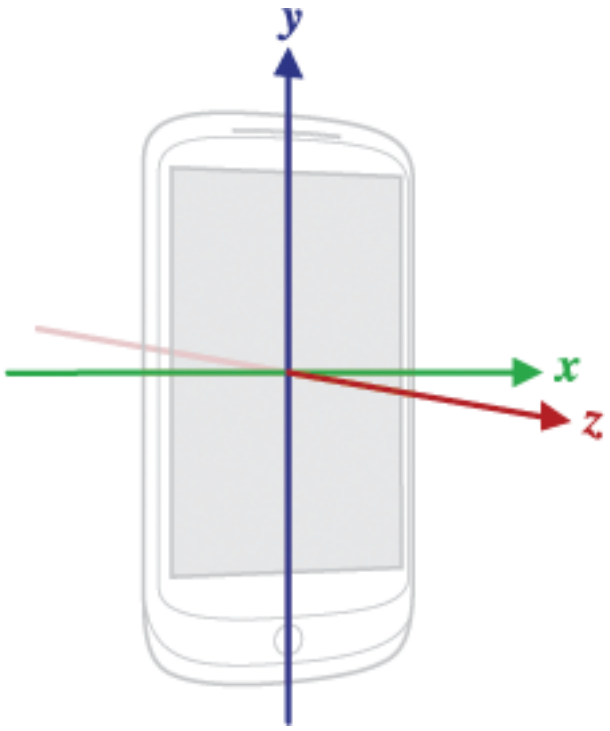

Рисунок 1.1 Стандартна орієнтація смарфтона у просторі, осі координат

Система Android дозволяє розробникам вказувати частоту дискретизації за допомогою однієї з чотирьох констант затримки сенсора:

1. SENSOR_DELAY_FASTEST
2. SENSOR_DELAY_GAME
3. SENSOR_DELAY_NORMAL
4. SENSOR_DELAY_UI

Виробники пристроїв можуть інтерпретувати ці значення по-різному. Процес фонового збору зразків для цього конкурсу вказував значення частоти дискретизації SENSOR_DELAY_NORMAL. Інші програми, які працюють на пристрої, можуть перезаписати вказану частоту дискретизації. Середній інтервал між послідовними зразками в цьому наборі даних становить 207 мс.

Сумарна сила, що діє на осі X, Y та Z пристрою, обчислюється за допомогою функції відстані: $\sqrt{(X^2 + Y^2 + Z^2)}$. Коли пристрій не прискорюється, ця функція оцінюється як сила гравітації (~9.85). Ця властивість гарантує, що

коли калібрований пристрій не рухається, координати акселерометра X, Y, Z, які він надає, утворюють точки на сфері з радіусом ~9.85.

На рисунку 1.2 можна побачити наступне - 670 зразків даних, зібраних з потоку акселерометра протягом приблизно 2 хвилин, коли пристрій тримався в альбомній, горизонтальній та вертикальній орієнтаціях.

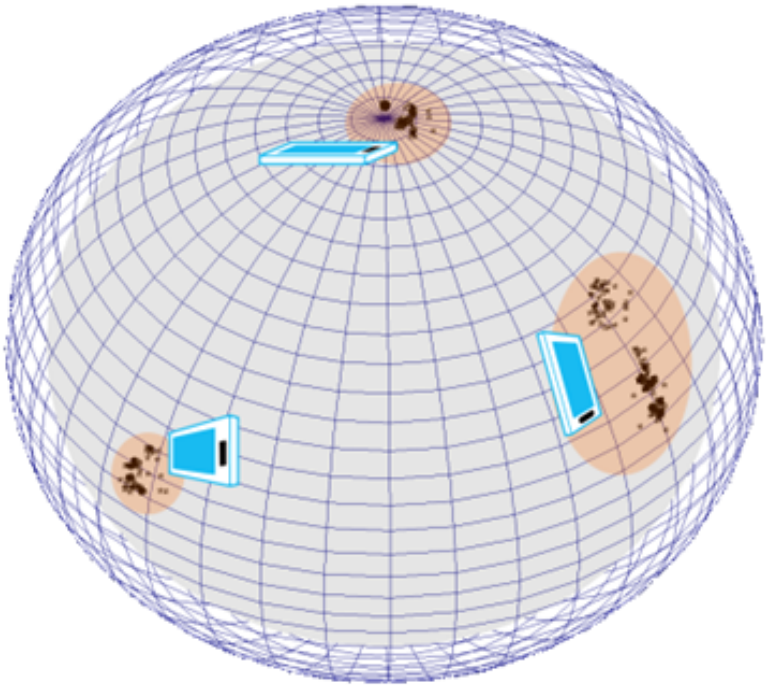

Рисунок 1.2 Візуалізація положення смартфону в тривимірному просторі впродовж якогось проміжку часу

Зверніть увагу, що зображені координати X, Y, Z кожної події акселерометра з'являються як точки на поверхні сфери.

1.4 Постановка задачі

У рамках даної дипломної роботи ставиться завдання розробки системи біометричної автентифікації користувача смартфона з використанням даних акселерометра. Основна ціль полягає у розробці швидкого, обчислювально-

дешевого та ефективного методу, який дозволить пасивно, у фоновому режимі ідентифіковувати користувача за його унікальними характеристиками руху, отриманими з акселерометра.

Джерелом набору даних, використаного у цій роботі, є відкритий набір даних акселерометра як біометричного ідентифікатора користувачів мобільних пристроїв на сайті Kaggle [5], який містить близько 60 млн рядків даних (часових рядів).

Ці часові ряди (сигнали) будуть конвертовані в скалеограми (scaleograms) за допомогою Вейвлет-перетворення (детальніше див. в розділі 2), і згодом будуть використані для навчання нейронної мережі на основі конволюцій (CNN).

Задача полягає у правильному перетворенні табличних даних у об’єкти зображень і, власне, створенні і навчанні оптимальної нейронної мережі, що включає в себе підбір оптимальних гіперпараметрів, експериментування з різними архітектурами, тощо.

Завершеною роботою є система, яка здатна автоматично визначати і класифікувати користувача смартфона за допомогою даних з його акселерометра, виводячи на екран ідентифікатор користувача. Розроблена система може бути використана для підвищення безпеки смартфонів та мобільних пристроїв, забезпечення якіснішого, ефективнішого та зручнішого користування пристроєм.

1.5 Висновки до розділу 1

У данному розділі була проведена обширна аналітична робота, спрямована на визначення актуальності задачі біометричної автентифікації користувача смартфона за допомогою даних акселерометра. В результаті ретельного огляду предметної області вдалося детально проаналізувати можливості використання акселерометра для біометричної автентифікації та виявити потенціал та

перспективність дослідження й розробки програмного забезпечення у цьому напрямку.

Джерелом набору даних, використаного у цій роботі, є відкритий набір даних акселерометра як біометричного ідентифікатора користувачів мобільних пристроїв на сайті Kaggle [5], який містить близько 60 млн рядків даних (часових рядів). Цей набір даних є публічно доступним і містить інформацію про акселерацію в трьох напрямках: X, Y та Z.

Ці дані дозволяють виконувати аналіз акселерометра та використовувати їх для розробки алгоритмів біометричної автентифікації.

РОЗДІЛ 2. МАТЕМАТИЧНІ ОСНОВИ РОБОТИ

2.1 Штучна нейронна мережа (ANN)

Нейронна мережа (також відома як штучна нейронна мережа) - це математична модель, інспірована роботою людського мозку. Вона складається з взаємопов'язаних вузлів, які називають нейронами, і зв'язків між ними, які передають сигнали (Рис. 2.1).

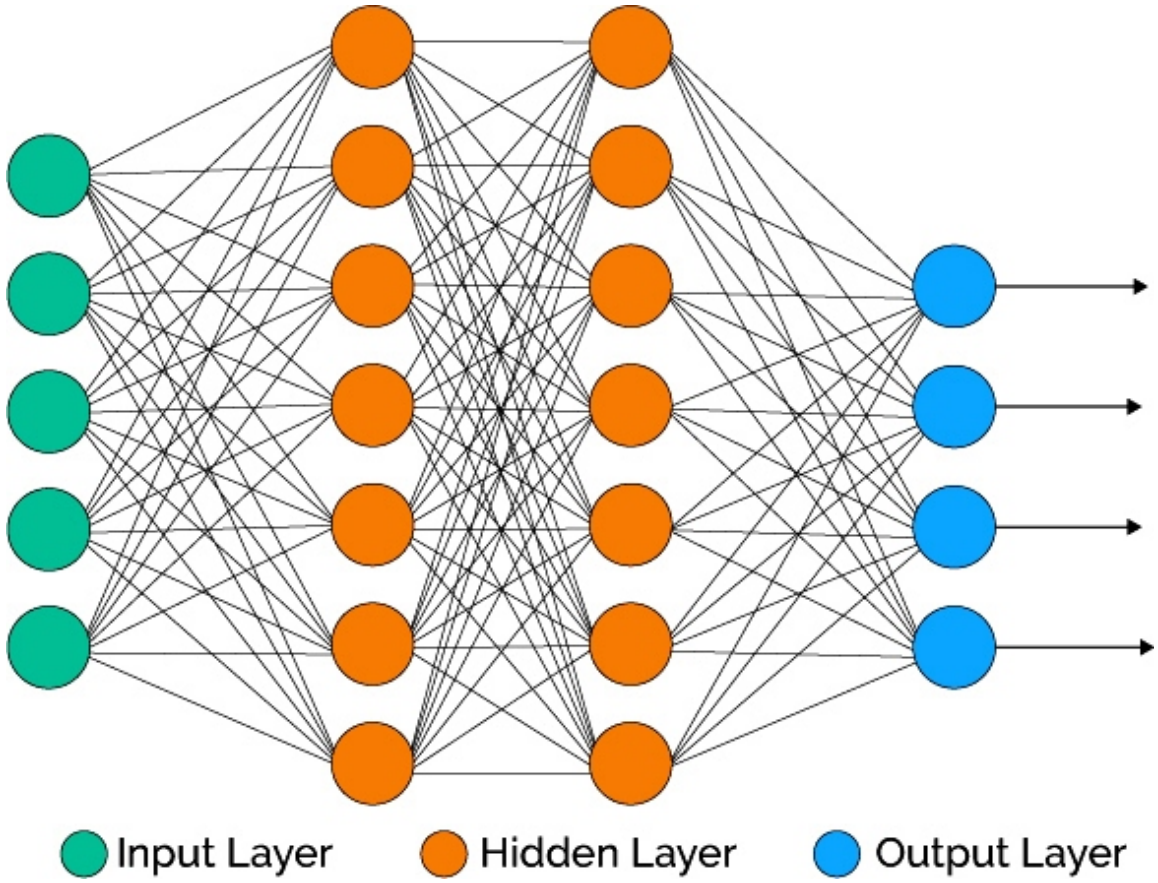

Рисунок 2.1 Схематичний вигляд нейронної мережі

Основна ідея полягає в тому, що нейрони приймають вхідні сигнали, обробляють їх та генерують вихідні сигнали [12]. Ці вхідні сигнали мають числове значення, яке може бути вагою, пов'язаною з кожним з'єднанням між нейронами. Нейрон обчислює зважену суму своїх вхідних сигналів і застосовує до неї активаційну функцію, яка визначає вихідний сигнал нейрона (Рис 2.2).

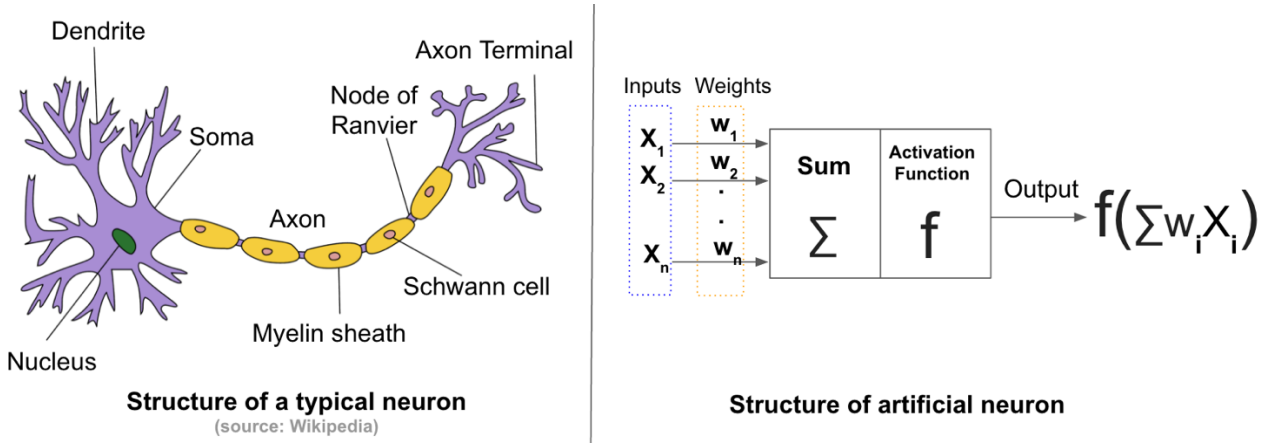

Рис. 2.2 Нейрон в людському мозку та математичне представлення нейрона в світі машинного навчання

Функції активації перетворюють зважену суму входів, які надходять на штучні нейрони [13]. Ці функції повинні бути нелінійними, щоб кодувати складні патерни даних. Найпопулярнішими функціями активації є Sigmoid, Tanh та ReLU. ReLU є найпопулярнішою функцією активації в глибоких нейронних мережах (Рис. 2.3).

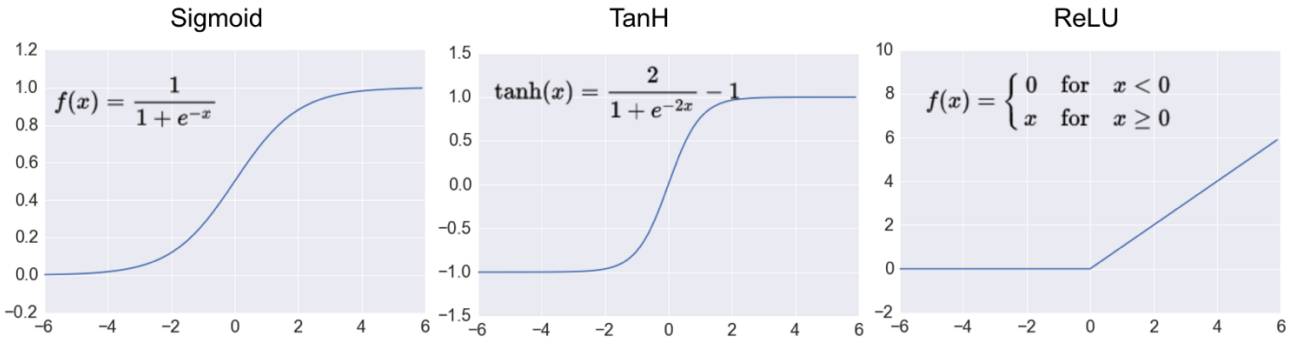

Рис 2.3 Поширені функції активації

Процес передачі сигналів через нейронну мережу можна уявити як каскадну передачу і обробку інформації. Вхідні дані подаються до вузлів у вигляді векторів, і ці вектори пропускаються через мережу, проходячи через

шари нейронів. Кожен шар мережі складається з набору нейронів, а вихідні сигнали одного шару передаються як вхідні сигнали наступного шару.

Процес передачі сигналів від початкового шару до кінцевого шару називається прямим проходженням (forward propagation). Після прямого проходження може бути застосована процедура поширення назад (backpropagation), яка використовується для підгонки параметрів мережі (наприклад, ваг) шляхом оцінки помилки між вихідними сигналами мережі і бажаними вихідними сигналами.

2.1 Recurrent Neural Network (RNN)

RNN розшифровується як Recurrent Neural Network - це тип штучної нейронної мережі, яка може обробляти послідовні дані, розпізнавати закономірності та прогнозувати кінцевий результат.

Ця нейронна мережа називається рекурентною, тому що вона може багаторазово виконувати одну і ту ж операцію над послідовністю вхідних даних.

RNN мають внутрішню пам'ять, яка дозволяє їм зберігати інформацію про попередні ітерації. Кожен вихідний шар RNN враховує попередній вихідний шар і передає його у наступну ітерацію. Це дозволяє RNN враховувати контекст та послідовність даних, що робить їх ефективними для завдань, які вимагають розуміння послідовностей, таких як мова, переклад, розпізнавання мови тощо [12].

Такі популярні продукти, як голосовий пошук Google і Siri від Apple, використовують RNN для обробки вхідних даних від своїх користувачів і прогнозування результатів.

Основний компонент RNN - це рекурентний шар (рис. 2.4), що складається з нейронів, які мають зв'язки з попереднім вихідним шаром. У кожній ітерації

RNN отримує вхідні дані і попередній вихідний шар. Вхідні дані можуть бути подані як один елемент послідовності на кожній ітерації або одразу у вигляді всієї послідовності.

Процес роботи RNN можна описати наступним чином:

1. На кожній ітерації вхідні дані та попередній вихідний шар надсилаються до рекурентного шару.
2. У рекурентному шарі відбувається обчислення нового внутрішнього стану (пам'яті) на основі вхідних даних і попереднього вихідного шару. Це здійснюється за допомогою зваженої суми вхідних значень та попереднього стану.
3. Новий внутрішній стан використовується для обчислення вихідного значення шару на поточній ітерації.
4. Отримане вихідне значення може використовуватися для подальшої обробки або передаватися до наступної ітерації як попередній вихідний шар.
5. Цей процес повторюється для кожного елемента послідовності, дозволяючи RNN враховувати контекст та залежності між елементами.

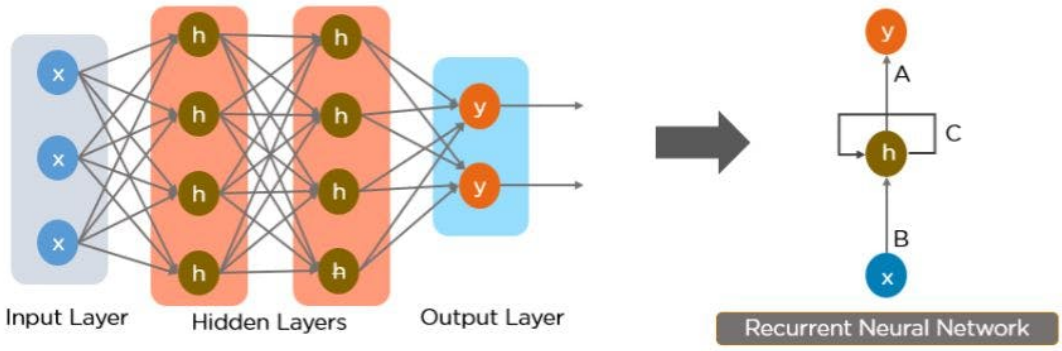

Рисунок 2.4 – Як можна перетворити нейронну мережу прямого поширення в рекурентну нейронну мережу

У лівій частині зображення зображено наступне:

X - вхідний шар.

h - прихований шар, який зберігає інформацію для попереднього виводу і подає її назад до себе.

y - вихідний шар

A, B, C - параметри для покращення результатів роботи моделі.

2.2 Long Short-Term Memory (LSTM)

Мережі з довгою короткочасною пам'яттю (LSTM) - це модифікована версія рекурентних нейронних мереж, яка полегшує запам'ятовування минулих даних у пам'яті [12]. Тут вирішується проблема зникаючого градієнта RNN. LSTM добре підходить для класифікації, обробки та прогнозування часових рядів з урахуванням часових лагів невідомої тривалості. Він навчає модель за допомогою зворотного поширення. У мережі LSTM присутні три входи (рис. 2.5):

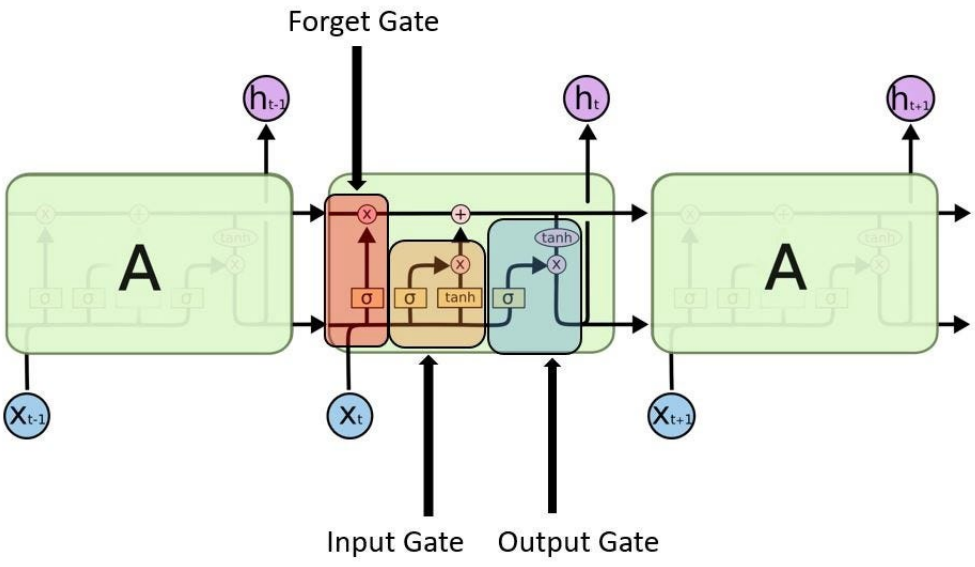

Рисунок 2.5 – Структура LSTM

1. Input Gate - визначають, яке значення з входу слід використовувати для модифікації пам'яті. Сигмоїдна функція вирішує, які значення пропускати, а функція tanh надає вагу значенням, що передаються, визначаючи їх рівень важливості в діапазоні від -1 до 1.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рисунок 2.6 – математичне підгрунтя Input Gate

2. Forget gate - визначають, які деталі слід викинути з блоку. Це вирішує сигмоїдна функція. Вона дивиться на попередній стан (ht-1) і вхідний вміст (Xt) і виводить число між 0 (опустити це) і 1 (залишити це) для кожного числа у стані комірки Ct-1:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рисунок 2.7 – математичне підгрунтя Forget Gate

3. Output Gate – вхідні значення і пам'ять блоку використовується для прийняття рішення про вихідні значення. Сигмоїдна функція вирішує, які значення пропускати через [0,1], а функція tanh надає вагу значенням, які передаються, визначаючи їх рівень важливості в діапазоні від -1 до 1 і множиться на вихід сигмоїдної функції.

$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рисунок 2.8 – математичне підгрунтя Output Gate

2.3 Convolutional Neural Networks (CNN)

CNN (Convolutional Neural Network) - це тип нейронної мережі, який спеціально призначений для обробки структурованих даних, зокрема зображень. Використовуючи згорткові шари та пулінг, CNN може автоматично виявляти та витягувати важливі ознаки зі вхідних даних, що допомагає розпізнавати шаблони, об'єкти та ієрархічні структури.

Основна логіка роботи CNN наступна:

1. Зображення вводиться у вхідний шар мережі.
2. Застосовуються різні фільтри (ядра), які виконують згортку зображення. Кожен фільтр шукає певну ознаку або шаблон у вхідних даних.
3. Після згортки використовується активаційна функція (зазвичай ReLU - Rectified Linear Unit) для створення нелинійності.
4. Застосовується процедура пулінгу (зазвичай MaxPooling), яка зменшує розмір зображення та вибирає найбільш важливі ознаки.
5. Кроки 2-4 повторюються для кількох шарів згортки та пулінгу, щоб глибше аналізувати ознаки.
6. Отримані ознаки передаються в повнозв'язаний шар (fully connected layer), де виконується класифікація або регресія відповідно до поставленої задачі.
7. Останній шар мережі дає вихідний результат, такий як класифікацію зображення або вектор значень для регресії.

Згорткові мережі були натхненні біологічними процесами у випадку патернів зв'язку нейронів. Вони також відомі як інваріантні до зсуву, оскільки базуються на архітектурі спільних ваг. На відміну від MLP (Multilayer perceptron), який використовує повністю пов'язаний шар (fully connected layer), CNN використовує різні шари для виявлення закономірностей серед зображень,

які передаються. CNN має мало зв'язаних шарів і приймає на вхід матрицю, тоді як MLP приймає на вхід вектори.

Як ви знаєте, кожне зображення складається з певної кількості пікселів. Ми аналізуємо вплив сусідніх пікселів на зображення, використовуючи так званий фільтр (його можна назвати ядром).

Фільтр - це тензор, який відстежує просторову інформацію і вчиться виокремлювати такі ознаки, як краї, згладженість кривих об'єкту, текстури у так званому згортковому шарі. Це допомагає відфільтрувати небажану інформацію для покращення зображень. Існують фільтри високих частот, де зміни інтенсивності відбуваються дуже швидко, наприклад, від чорного до білого пікселя і навпаки.

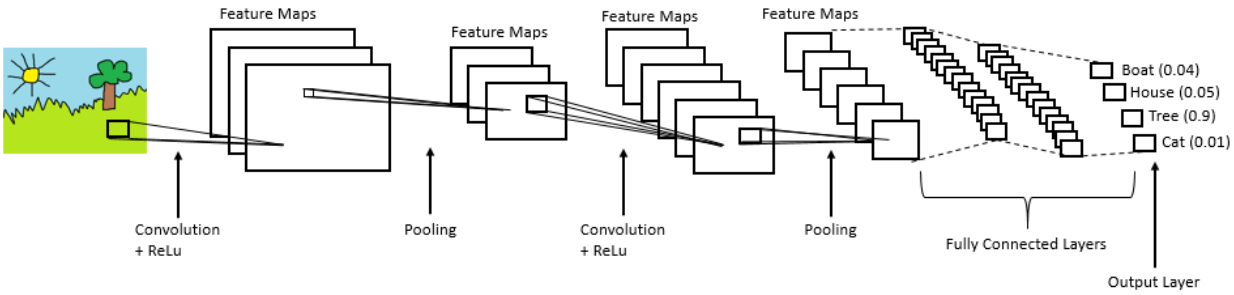

Рисунок 2.9 – Принцип роботи CNN

На схемі вище ми можемо бачити:

1. Шар згортки, де відбувається згортка.
2. Шар об'єднання, де відбувається процес об'єднання
3. Нормалізація, зазвичай з використанням ReLu
4. Повністю з'єднані шари

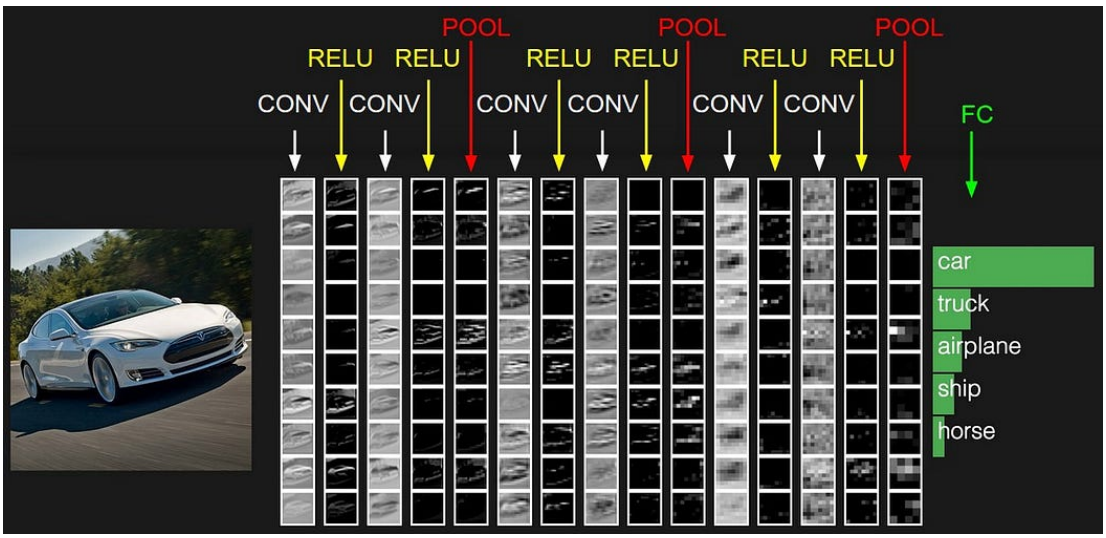

Рисунок 2.10 – Виявлені нейронною мережею обриси об’єкта

CNN ефективно використовуються для розпізнавання образів, класифікації зображень, визначення об'єктів на зображенні, детекції облич, сегментації зображень, аналізу відео, генеративних моделей зображень та багатьох інших завдань комп'ютерного зору. Вони також можуть застосовуватися в обробці природної мови, зокрема для завдань, пов'язаних з обробкою тексту та послідовностей.

Отже, CNN можна використовувати для обробки послідовностей. Але в порівнянні з LSTM (Long Short-Term Memory) та RNN (Recurrent Neural Network) вони мають деякі обмеження. Основна перевага LSTM та RNN полягає у їх здатності до моделювання довготривалих залежностей в послідовностях, особливо в тих, де попередні дані можуть впливати на майбутні результати.

Ось деякі важливі відмінності між CNN та LSTM/RNN:

1. Моделювання залежностей у часі: LSTM та RNN розроблені спеціально для роботи з послідовними даними і можуть ефективно моделювати довготривалі залежності в часі. Вони зберігають стан попередньої ітерації та передають його наступній ітерації, що дозволяє їм утримувати інформацію про попередні дані.

2. Локальні шаблони та просторові ознаки: CNN краще пристосовані для виявлення локальних шаблонів та просторових ознак у вхідних даних, таких як зображення. Вони використовують згорткові шари для локального збору інформації і пулінгові шари для виділення найважливіших ознак. Однак вони можуть пропустити деякі довготривалі залежності в послідовностях, які LSTM та RNN здатні відслідковувати.

3. Розмір вхідних даних: CNN найкраще працюють з фіксованими розмірами вхідних даних, такими як зображення, де ширина та висота можуть бути задані наперед. LSTM та RNN можуть обробляти послідовності різної довжини, що робить їх більш гнучкими для різних типів послідовних даних.

Отже, якщо маєте справу з послідовностями, особливо з тривалими залежностями у часі, LSTM та RNN можуть бути кращим варіантом. Однак, якщо ви маєте справу з обробкою зображень або вам потрібно виявити локальні шаблони та просторові ознаки, то CNN будуть більш ефективними. В деяких випадках комбінація цих двох підходів, наприклад, використання CNN для виділення ознак та LSTM/RNN для аналізу послідовностей, може бути найкращим рішенням.

2.4 Оптимізатори

Алгоритми оптимізації сприяють покращенню ефективності моделей глибокого навчання шляхом значного прискорення тренування та підвищення його точності. Під час тренування моделі важливо досягти мінімуму функції втрат і оновлювати ваги на кожній епосі. Оптимізатор є алгоритмом або функцією, які змінюють параметри, такі як ваги та швидкість навчання, з метою зниження загальної втрати (Loss) і покращення точності моделі.

2.4.1 Градієнтний спуск

Цей алгоритм оптимізації систематично модифікує значення ваг та досягає локального мінімуму за допомогою математичних обчислень. Формула (2.1) описує те, що робить алгоритм градієнтного спуску [14].

$$b = \alpha - \gamma \nabla f(\alpha) \tag{2.1}$$

b - наступна позиція, а α - поточна позиція. Мінус перед знаком відповідає мінімізації. Гама посередині - це коефіцієнт ваги, а градієнтна частина - просто напрямок найшвидшого спуску. Таким чином, ця формула показує нам наступну позицію, яка вказує напрямок найшвидшого спуску.

Градієнтний спуск розпочинається з початкових коефіцієнтів, обчислює їх значення та шукає меншу вагу. Потім він рухається в напрямку меншої ваги і оновлює значення коефіцієнтів. Цей процес повторюється до досягнення локального мінімуму. Локальний мінімум представляє собою точку, в якій неможливо продовжити рух [14]. Градієнтний спуск є ефективним підходом для багатьох завдань, однак, у нього є певні обмеження. При великому обсязі даних обчислення градієнту можуть бути витратними з точки зору часу та ресурсів. Крім того, для неконвексних функцій, де можуть існувати багато локальних екстремумів або немає жодного екстремуму, градієнтний спуск може бути менш ефективним рішенням.

2.4.2 Стохастичний градієнтний спуск

Як зазначалось раніше, градієнтний спуск на великих наборах даних може бути далеко не найкращим вибором оптимізатора. Звернімо увагу на стохастичний градієнтний спуск. Цей алгоритм базується на випадковості, що і означає термін "стохастичний". Замість обчислення градієнта на всіх даних, ми випадковим чином вибираємо підмножину даних, яку називають пакетом (batch).

Формула (2.2) описує кроки, які виконуються в стохастичному градієнтному спуску (SGD). Спочатку ми ініціалізуємо параметри (ваги), позначені як w, і швидкість навчання, позначену як η. Потім на кожній ітерації дані випадковим чином перемішуються, щоб досягти приблизної мінімальної точки [14].

$$w = w - \eta \nabla Q_i(w) \tag{2.2}$$

Оскільки ми використовуємо лише частину даних у кожній ітерації, шлях, яким рухається алгоритм, містить більше шуму порівняно з алгоритмом градієнтного спуску. Тому SGD потребує більшої кількості ітерацій, щоб досягти локального мінімуму. Зі збільшенням числа ітерацій збільшується і загальний час обчислення. Проте, навіть після збільшення числа ітерацій, ресурсозатратність все ще нижча, ніж у класичного градієнтного спуску.

2.4.3 Adagrad (Адаптивний градієнтний спуск)

Алгоритм адаптивного градієнтного спуску відрізняється від інших алгоритмів градієнтного спуску тим, що використовує різні швидкості навчання для кожної ітерації. Зміна швидкості навчання залежить від різниці у параметрах під час тренування. Ця модифікація є дуже корисною, оскільки реальні набори даних містять як розріджені, так і щільні ознаки, і недопустимо мати одне значення швидкості навчання для всіх ознак.

Формули (2.3) і (2.4) показують, як алгоритм Adagrad оновлює ваги. Тут α_t позначає різні швидкості навчання на кожній ітерації, η - це константа, а ϵ - мала позитивна величина, що додається для уникнення ділення на 0 [14].

$$W_t = W_{t-1} - \eta'_t \frac{\partial L}{\partial w(t-1)} \tag{2.3}$$

$$\eta_t' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}} \quad (2.4)$$

Використання оптимізатора Adagrad має важливу особливість - він усуває потребу вручну налаштовувати швидкість навчання. Цей метод збігається до мінімуму швидше і є надійнішим порівняно з алгоритмами градієнтного спуску та іншими варіаціями.

Однак основним недоліком оптимізатора Adagrad є його можливість зменшення швидкості навчання до дуже малих значень на певному етапі. Це може негативно вплинути на точність моделі, оскільки модель не здатна отримувати нові знання. Це може статися через накопичення квадратів градієнтів у знаменнику, що приводить до зростання значення знаменника [14].

2.4.4 RMS Prop (Roor Mean Square)

Алгоритм RMS prop є розширеною версією алгоритму RProp (resilient backpropagation) [14], який вирішує проблему змінних градієнтів. Проблема полягає в тому, що деякі градієнти можуть бути великими, тоді як інші - малими. Тому визначення однієї швидкості навчання може бути не найкращим варіантом. У алгоритмі RProp спочатку порівнюються два градієнти за їхніми знаками. Якщо вони мають однакові знаки, то крок збільшується на невелику частку, оскільки ми рухаємося у правильному напрямку. Якщо вони мають протилежні знаки, то крок зменшується. Крок потім обмежується, і ваги оновлюються відповідно.

Однак проблема з алгоритмом RProp полягає в тому, що він має складності у роботі з великими наборами даних. Тому був розроблений алгоритм RMS prop, який іноді вважається поліпшенням оптимізатора AdaGrad, оскільки зменшує монотонно спадаючу швидкість навчання [14].

Основний фокус алгоритму RMS prop - прискорити процес оптимізації, зменшивши кількість обчислень функцій, необхідних для досягнення локального мінімуму. Алгоритм зберігає крайнє значення квадратів градієнтів для кожної ваги і ділить градієнт на корінь середньоквадратичного значення. Формула (2.5) математично описує цей процес.

$$v(w,t) = \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \tag{2.5}$$

Де γ - це коефіцієнт забуття. Ваги оновлюються за формулою (2.6).

$$w = w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w) \tag{2.6}$$

Якщо існує параметр, через який функція витрат сильно коливається, ми хочемо уникнути оновлення цього параметра. Цей алгоритм має кілька переваг порівняно з попередніми версіями алгоритмів градієнтного спуску. Він швидко збігається і вимагає менше налаштувань, ніж алгоритми градієнтного спуску та його варіації.

Проблема з RMS Prop полягає в тому, що коефіцієнт навчання має бути визначений вручну, і запропоноване значення не підходить для кожної задачі [14].

2.4.5 Adam

Adam, що походить від "адаптивної оцінки моменту", є модифікацією стохастичного градієнтного спуску, який використовується для оновлення ваг мережі під час навчання.

Одним з відмінних рис Adam від стохастичного градієнтного спуску є окреме оновлення коефіцієнта навчання для кожної ваги. Крім того, Adam поєднує переваги алгоритмів AdaGrad і RMSProp. Алгоритм RMSProp адаптує

коефіцієнти навчання на основі середньоквадратичного градієнту, тоді як Adam використовує математичне очікування квадратичних відхилень від середнього, що також називається некоректованою дисперсією.

Формули (2.7) та (2.8) представляють роботу оптимізатора Adam. Тут β_1 та β_2 відображають частку затухання середнього градієнту [14].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\left[\frac{\delta L}{\delta w_t}\right] \tag{2.7}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2 \tag{2.8}$$

Оптимізатор Adam володіє кількома перевагами, що робить його широко використовуваним та рекомендованим як алгоритм оптимізації за замовчуванням. Його простота у реалізації, швидкий час роботи, низький обсяг використовуваних ресурсів та менша потреба в налаштуванні порівняно з іншими алгоритмами оптимізації робить його привабливим для багатьох задач.

Проте, навіть у Adam є деякі недоліки, які варто враховувати. Він має схильність до швидших обчислень, що може бути неважливим фактором, особливо в порівнянні з алгоритмами, такими як стохастичний градієнтний спуск, які акцентують увагу на самому процесі обробки даних. Іншими словами, алгоритми, подібні до SGD, можуть забезпечувати кращу узагальненість даних завдяки меншому обчислювальному зусиллю. Тому при використанні Adam варто враховувати цю особливість та вибрати оптимальний алгоритм оптимізації залежно від конкретної задачі та обмежень ресурсів [14].

2.5 Метрики

Я використовую набір стандартизованих метрик ефективності для оцінки роботи моїх моделей в цьому дослідженні. Отже, використовуються такі метрики:

- Частка помилкового прийняття (False acceptance rate, FAR) - це ймовірність класифікації сегментів від шахрайського користувача як від аутентичного користувача. Частка помилкового прийняття обчислюється за формулою: FAR = p(authentic|fraudulent)
- Accuracy = (Кількість правильно класифікованих зразків) / (Загальна кількість зразків)
- Частка помилкового відхилення (False rejection rate, FRR) - це ймовірність класифікації сегментів від аутентичного користувача як від шахрайського користувача. Частка помилкового відхилення обчислюється за формулою: FRR = p(fraudulent|authentic)
- Рівень рівноцінних помилок (Equal error rate, EER) - це відсоток, при якому частка помилкового відхилення дорівнює частці помилкового прийняття. Це загальна метрика, яка використовується для оцінки ефективності нашої моделі.
- Крива характеристики роботи приймача (Receiver operating characteristic, ROC) - це крива, яка використовується для представлення ефективності бінарного класифікатора. На цій криві відображається частка правильного прийняття, тобто 1-FRR, як функція частки помилкового прийняття.
- Площа під кривою (Area under the curve, AUC) - це площа під кривою ROC. Вона використовується для оцінки ефективності класифікатора. Чим більше значення площі під кривою, тим краща ефективність класифікатора. Використовуючи ці метрики, ми можемо детально оцінити ефективність нашої моделі при виконанні різних завдань.

2.6 Перетворення Фур'є

Що, якщо ми просто введемо сирі сигнали (timeseries) в нейронну мережу по типу LSTM, яка здатна робити багатокласову класифікацію? Здавалося б, найпростіше рішення проблеми!

Виявляється, не все так просто. Уявіть, що у нас є дані двох людей, які йдуть в одному темпі, але їхні кроки не синхронізовані. Ми можемо змоделювати це, взявши один сигнал і зсунувши його трохи вперед у часі, щоб створити інший сигнал. Інтуїтивно, якщо ми подивимось на обидва графіки, то зрозуміємо, що якщо перший представляє ходьбу, то другий, ймовірно, теж. Але нейронна мережа не має такої ж інтуїції. Порівнюючи два графіки точка за точкою, нейромережа виявить великі відмінності в кожному з них. Тому для мережі не буде одразу очевидно, що сигнали представляють одну і ту ж активність.

Здається, якби ми могли знайти спосіб ігнорувати точний час зміни сигналу, але зберегти інформацію про форму сигналу (частоту та амплітуду), ми могли б вирішити цю проблему. Виявляється, для цієї проблеми вже існує математичне рішення – і це перетворення Фур'є.

Перетворення Фур'є - це математична операція, яка використовується для розкладання сигналу на його складові частоти. Вона перетворює сигнал з часової області в частотну область, що дозволяє аналізувати його спектральні властивості.

Основна ідея перетворення Фур'є полягає в тому, що будь-який складний сигнал може бути розкладений на суму простіших сигналів, які мають різні частоти (рис. 2.11). Ці простіші сигнали називаються гармоніками або частотними компонентами. Кожна гармоніка відповідає певній частоті і має свою амплітуду і фазу.

Перетворення Фур'є можна застосовувати як до дискретних, так і до неперервних сигналів. Для дискретних сигналів використовується дискретне перетворення Фур'є (DFT), а для неперервних сигналів - неперервне

перетворення Фур'є (FT). В практичних застосуваннях найчастіше використовується швидке перетворення Фур'є (FFT), яке є ефективним алгоритмом обчислення DFT.

Процес перетворення Фур'є можна уявити як розкладання сигналу на суму синусоїд і косинусоїд з різними амплітудами і фазами. Цей розклад дозволяє отримати інформацію про спектральні складові сигналу, тобто про те, які частоти присутні у сигналі і з якою силою.

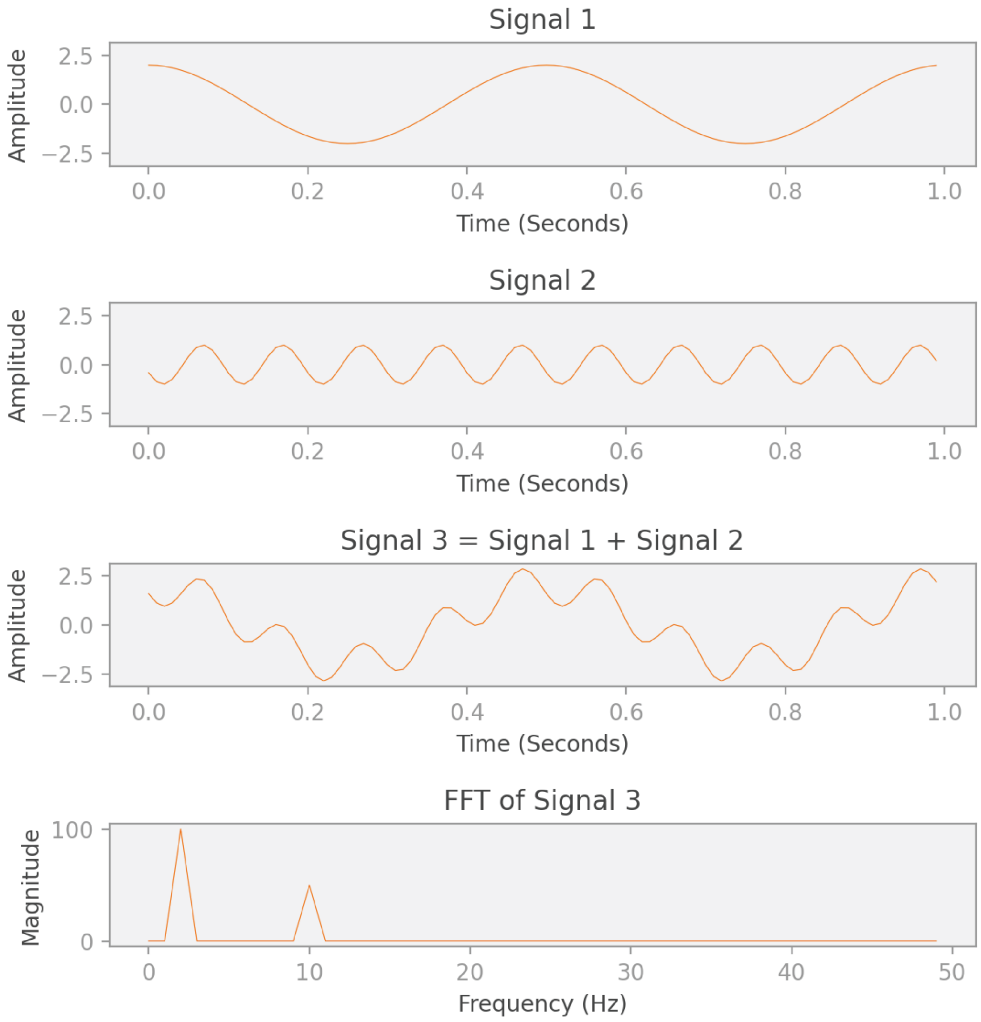

Рисунок 2.11 Приклад Швидкого Перетворення Фур’є (FFT)

Як ви можете бачити на зображенні, FFT сигналу 3 дає нам функцію від частоти. Форма цього FFT є цікавою: вона має піки на 2 Гц і на 10 Гц. Це говорить нам про те, що сигнал 3 можна розкласти на два інших сигнали, один з яких коливається з частотою 2 Гц, а інший - з частотою 10 Гц. Ми знаємо, що

це правда, тому що саме так ми побудували сигнал 3. Зауважте також, що величина піку на частоті 10 Гц вдвічі менша за величину піку на частоті 2 Гц, що відображає відносну амплітуду двох вихідних сигналів.

Це саме та інформація, яку ми шукали: опис сигналу з точки зору його форми, ігноруючи часові та фазові відмінності. Цікавим у перетворенні Фур'є є те, що його можна обчислити для (майже) будь-якого сигналу - навіть якщо сигнал не є періодичним, його можна розкласти у зважену суму періодичних функцій (синусів і косинусів), які його апроксимують.

Тепер подивимось на формулу перетворення Фур'є для функції (2.9):

$$\mathcal{F}(f(t)) = F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt \tag{2.9}$$

Як бачимо, перетворення Фур'є є функцією *F* кутових частот *w* - для кожної частоти воно дає єдине значення, яке агрегує інформацію для всієї області вихідного сигналу *f(t)*.

Погляньмо на ще один приклад. Перед вами випадковий сигнал з мого датасету (рис. 2.12):

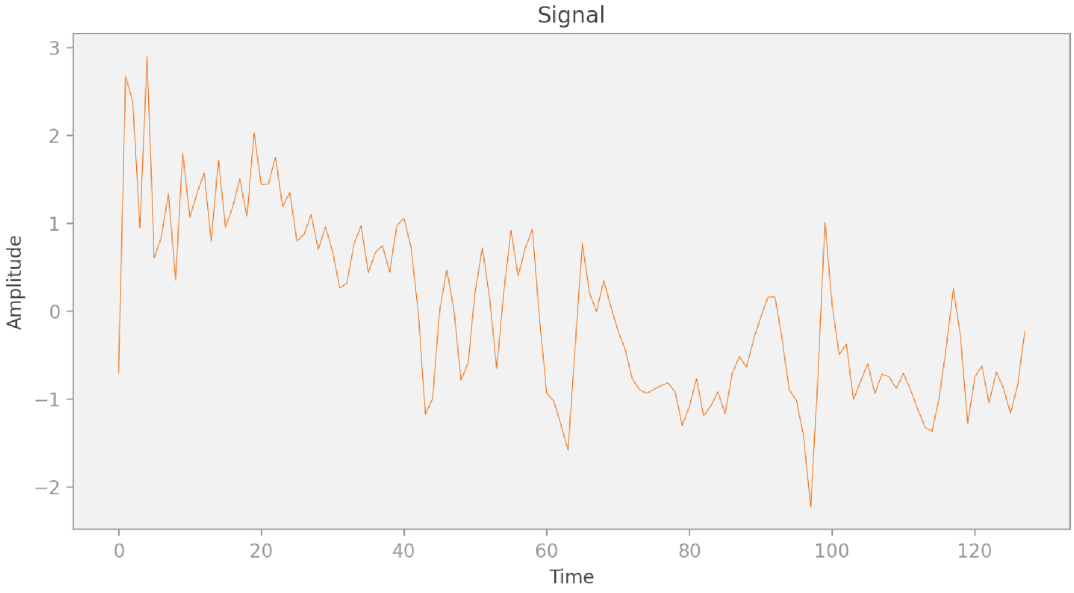

Рисунок 2.12 «Сирий» сигнал

А тепер обчислимо FFT сигналу (рис. 2.13):

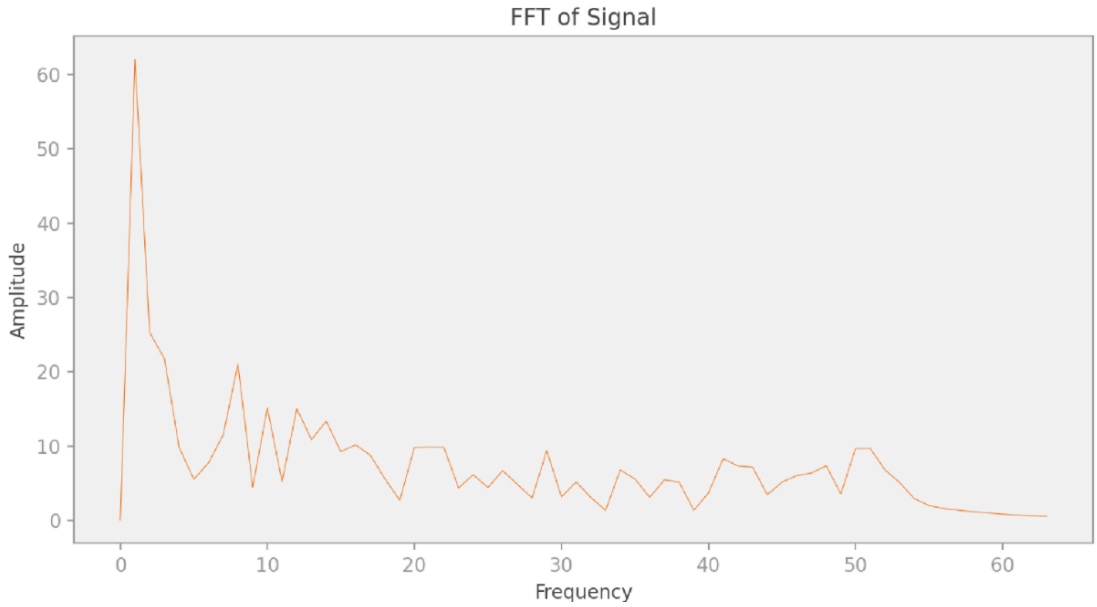

Рисунок 2.13 Результат застосування FFT на сигнал з рис. 2.12

Оскільки форма цього вхідного сигналу набагато складніша, ніж у попередньому простому прикладі, нам потрібно скласти набагато більше періодичних сигналів різної частоти, щоб правильно його представити. Але ми все одно можемо це зробити, навіть якщо сигнал не є періодичним.

Перетворення Фур'є має широке застосування в різних галузях, зокрема в сигнальній обробці, обробці зображень, телекомунікаціях, акустичному аналізі, медичній діагностиці, комп'ютерній графіці і багатьох інших. Використовуючи перетворення Фур'є, можна виявити характерні частоти сигналу, виділити сигнали на фоні шуму, виконувати фільтрацію сигналів, стиснення даних та багато іншого.

2.7 Перетворення Габора (STFT)

Однією зі стратегій збереження як частоти, так і часу є використання перетворення Габора, яке також називають короткочасним перетворенням Фур'є (STFT). Перетворення Габора обчислює FFT на рухомому вікні часу і може бути

візуалізоване за допомогою графіка амплітуди як функції частоти і часу, який ми називаємо спектрограмою.

При обчисленні перетворення Габора ми виконуємо наступні операції для часу t від 0 до n (у нашому сценарії n=128):

- Центруємо функцію-фільтр g в момент часу g. Зазвичай ми використовуємо гауссову функцію як фільтр, як показано на зображенні нижче, але замість неї можна використовувати й інші фільтри.
- Ми перемножуємо вихідний сигнал f і фільтр g для всіх значень τ, в результаті чого отримуємо відфільтрований сигнал тієї ж довжини, що і вихідний сигнал. Цей відфільтрований сигнал зберігає інформацію вихідного сигналу в інтервалі часу t, а решту інформації ігнорує.
- Нарешті, ми беремо FFT відфільтрованого сигналу, який показує частоти, присутні протягом проміжку часу, виділеного гауссовим фільтром.

Я продемонструю цю ідею для відрізків t=20 і t=100 на рисунку 2.14:

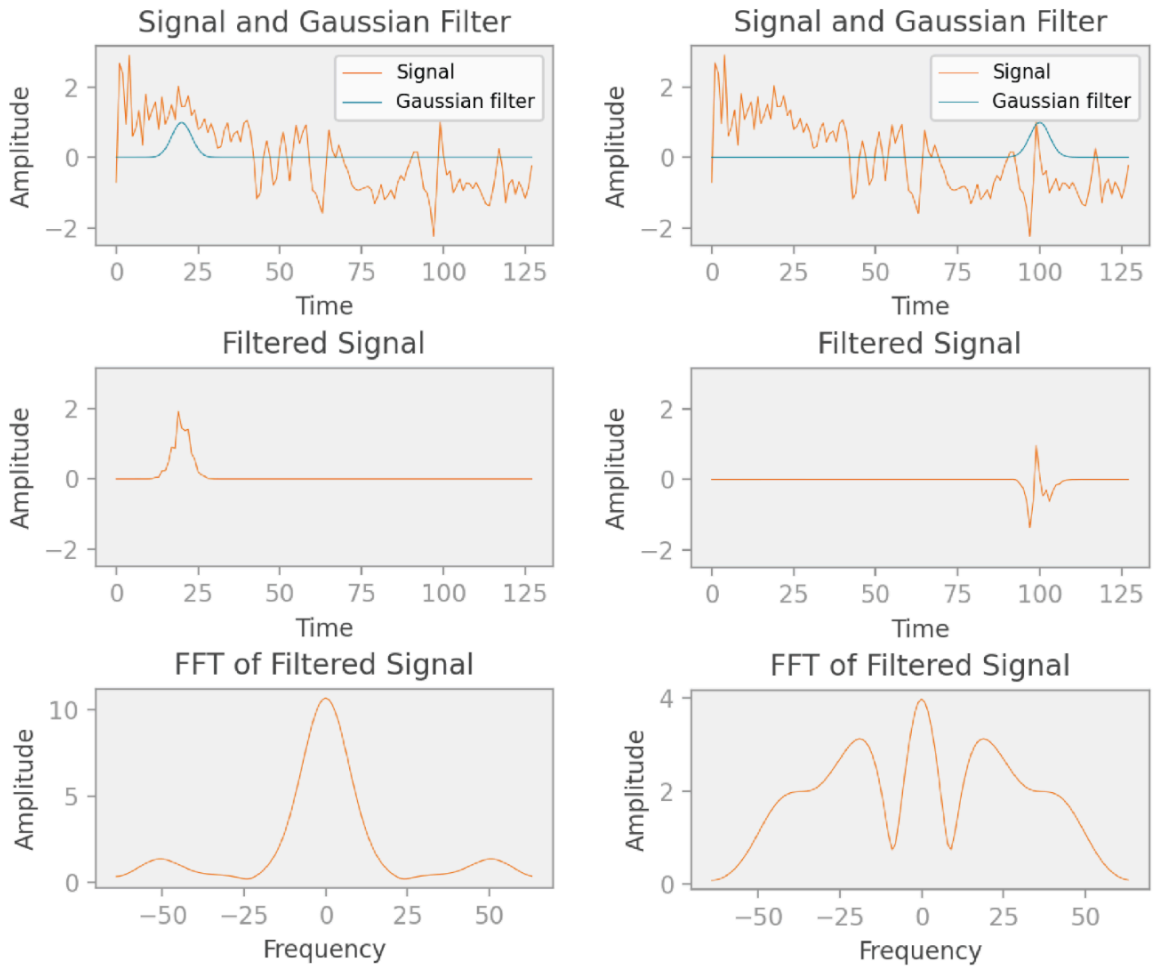

Рисунок 2.14 Приклад застосування фільтр-функції та FFT

Ви могли помітити, що графіки FFT, показані тут, мають як позитивні, так і негативні частоти і є парними (симетричними) функціями, в той час як графік FFT, показаний в попередньому розділі, мав лише позитивні частоти. Насправді, графік FFT у попередньому розділі виглядає подібно до наведеного тут, але я вирішив показати лише додатні частоти, щоб зосередитися на поясненні основних концепцій. Точніше кажучи, FFT складається з комплексних чисел, і якщо вихідний сигнал є дійсним (як у нашому випадку), то відповідне FFT завжди є "спряженим симетричним". Це означає, що його дійсна частина є парною функцією (віддзеркаленою по осі абсцис), а уявна частина - непарною функцією (віддзеркаленою по осі абсцис, а потім по осі ординат). Зараз я замість того, щоб розглядати дійсну та уявну частини окремо, візьму магнітуду (magnitude) кожного комплексного числа, яке повертає FFT, що також

виявляється парною функцією. Я використовую магнітуду, тому що нам знадобляться дійсні значення для наступного кроку, де ми будемо візуалізувати перетворення Габора у вигляді зображення спектрограми.

Після того, як ми розмістили функцію Гаусса g в кожному часовому значенні t і завершили обчислення перетворення Габора, ми отримуємо 128 FFT, кожне з яких має 128 значень, як і вихідний сигнал. Таким чином, ми отримуємо 128 × 128 індивідуальних значень. Ці значення можна представити у вигляді зображення, яке називається спектрограмою, як показано нижче (рис. 2.15):

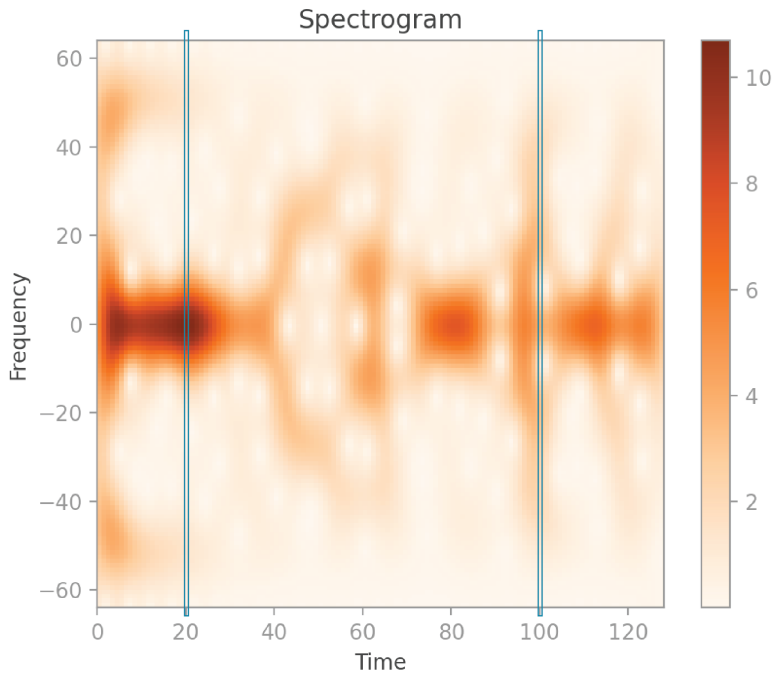

Рисунок 2.15 Спектрограма

Ось хороший спосіб візуалізувати, як було створено цю спектрограму: при t=0, ви виконуєте послідовність обчислень, описану раніше, і отримуєте назад 128 амплітуд, обчислених за допомогою FFT. Кожне з цих значень можна представити у вигляді пікселя на спектрограмі, причому яскравість пікселя відповідає амплітуді. Ви поміщаєте цю послідовність пікселів у крайній лівий стовпчик спектрограми, що відповідає часу 0. Потім переходите до t=1, виконуєте ті ж самі обчислення і заповнюєте пікселі в наступному стовпчику в момент часу 1. І так далі, для всіх 128 моментів часу.

Зверніть увагу, що спектрограма вертикально симетрична. Це тому, що кожен стовпчик містить величину значень FFT, який, як ми бачили раніше, є парною функцією. Я виділив стовпчики з частотами 20 і 100, які відповідають розрахункам, що ми бачили на попередніх графіках. Якщо ви уважно подивитеся, то помітите, що частоти з більшою амплітудою на графіках FFT відповідають темнішим пікселям на спектрограмі.

Нижче наводжу повну формулу для перетворення Габора (2.10):

$$\mathcal{G}(f(t)) = G(t, \omega) = \int_{-\infty}^{\infty} f(\tau)\, \bar{g}(\tau - t)\, e^{-i\omega\tau}\, d\tau \tag{2.10}$$

Спектрограми є чудовим способом візуалізації частот, присутніх у сигналі в часі. Однак вони мають суттєвий недолік: як тільки ви виберете значення стандартного відхилення в гауссовому фільтрі, гауссовий фільтр буде найефективнішим для виявлення певної частоти, і гіршим для виявлення набагато нижчих частот і для локалізації набагато вищих частот.

Уявіть, що одна з частот, з яких складається ваш вихідний сигнал, є дуже широкою хвилею (яка представляє дуже низьку частоту). Якщо гаусівський фільтр набагато вужчий за цю хвилю, він не зможе відфільтрувати і виявити цю низьку частоту в жодному з часових вікон. Одним із способів пом'якшити цю проблему є використання широкого гауссового фільтра. Однак з широким фільтром ми зберігаємо менше часової інформації, оскільки наше ковзне вікно набагато більше. Отже, це компроміс: якщо нас турбує збереження низьких частот, ми обираємо широкий фільтр; якщо нас турбує збереження часової інформації, ми обираємо вузький фільтр.

2.8 Вейвлет-перетворення (Wavelet transform)

Вейвлети фіксують частоти в часі так само, як і перетворення Габора, але вони не вимагають від нас компромісів щодо точності нашої часової інформації або діапазону частот, які вони виявляють. Так само, як перетворення Габора можна візуалізувати за допомогою спектрограми, вейвлет-перетворення можна візуалізувати за допомогою скалеограми.

Механізм, що використовується у вейвлет-перетворенні, дещо відрізняється від перетворення Габора. У перетворенні Габора ми множимо вихідний сигнал на гаусівський фільтр, який транслюється в часі. У вейвлет-перетворенні ми множимо вихідний сигнал на вейвлет - за винятком того, що вейвлет не тільки транслюється в часі, але й масштабується по ширині.

Дрібномасштабні вейвлети дозволяють нам захоплювати високі частоти в межах точних часових інтервалів, а великомасштабні вейвлети дозволяють нам захоплювати низькі частоти на довших часових інтервалах.

При обчисленні вейвлет-перетворення ми виконуємо наступні операції для кожного часу t від 0 до n, і для різних n значень s масштабу (де n=128 в нашому випадку):

- Масштабуємо вейвлет $\boldsymbol{\psi}$ відповідно до параметра s і центруємо його в момент часу t. Вейвлет, показаний на графіках нижче, є «вейвлетом мексиканського капелюха» («Mexican hat wavelet»), але існує багато інших широко використовуваних вейвлетів, які ми могли б вибрати замість нього.
- Ми перемножуємо вейвлет і сигнал разом, щоб отримати відфільтрований сигнал.
- Ми обчислюємо інтеграл (або суму, в дискретному випадку) відфільтрованого сигналу. Отже, кінцевий результат - це скаляр.

Рисунок 2.16 нижче ілюструє цю ідею. Графіки у верхньому ряду ілюструють два перетворення вузького вейвлету, а графіки в нижньому ряду ілюструють два перетворення ширшого вейвлету.

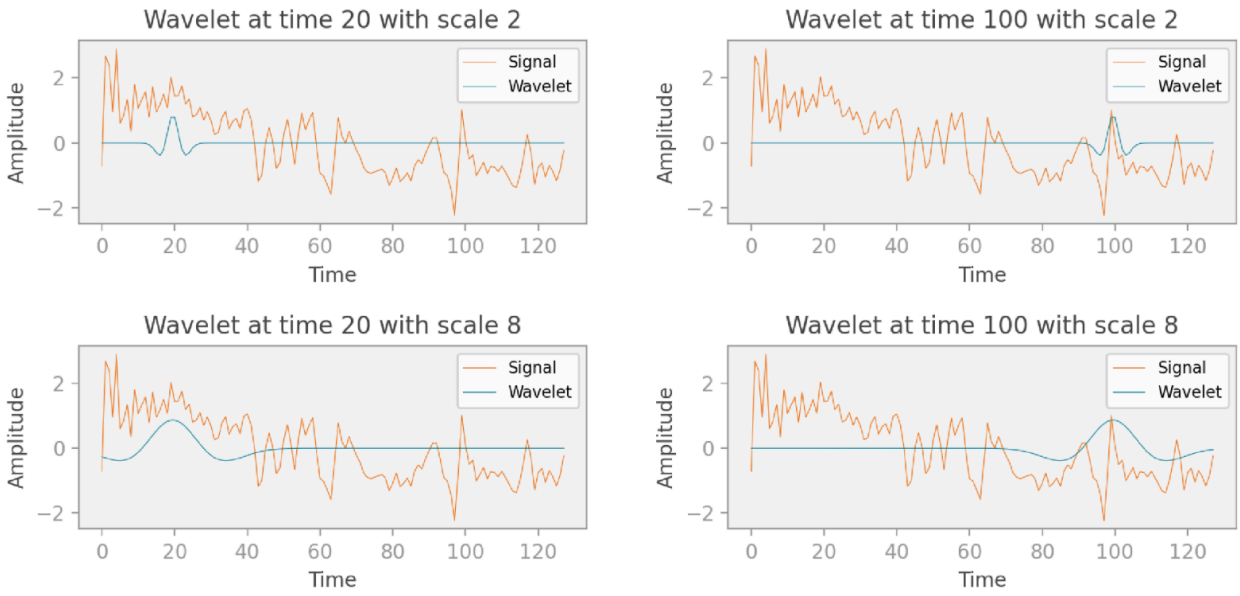

Рисунок 2.16 Різні вейвлет перетворення

Тепер давайте подумаємо про розміри скалеограми, яку ми створимо для візуалізації цих обчислень. Як я вже згадував, кінцевим результатом одного проходу обчислень є скаляр. Ми виконуємо ці обчислення для *n* перекладів і масштабів (scales). Отже, скалеограма має розміри *n***n*, і її можна представити у вигляді двовимірного зображення. На скалеограмі нижче (рис. 2.17) я виділив чотири пікселі - вони відповідають чотирьом графам на попередньому малюнку. Ви можете бачити, що це значення, отримані при часі 20 і 100, а також при масштабах 2 і 8.

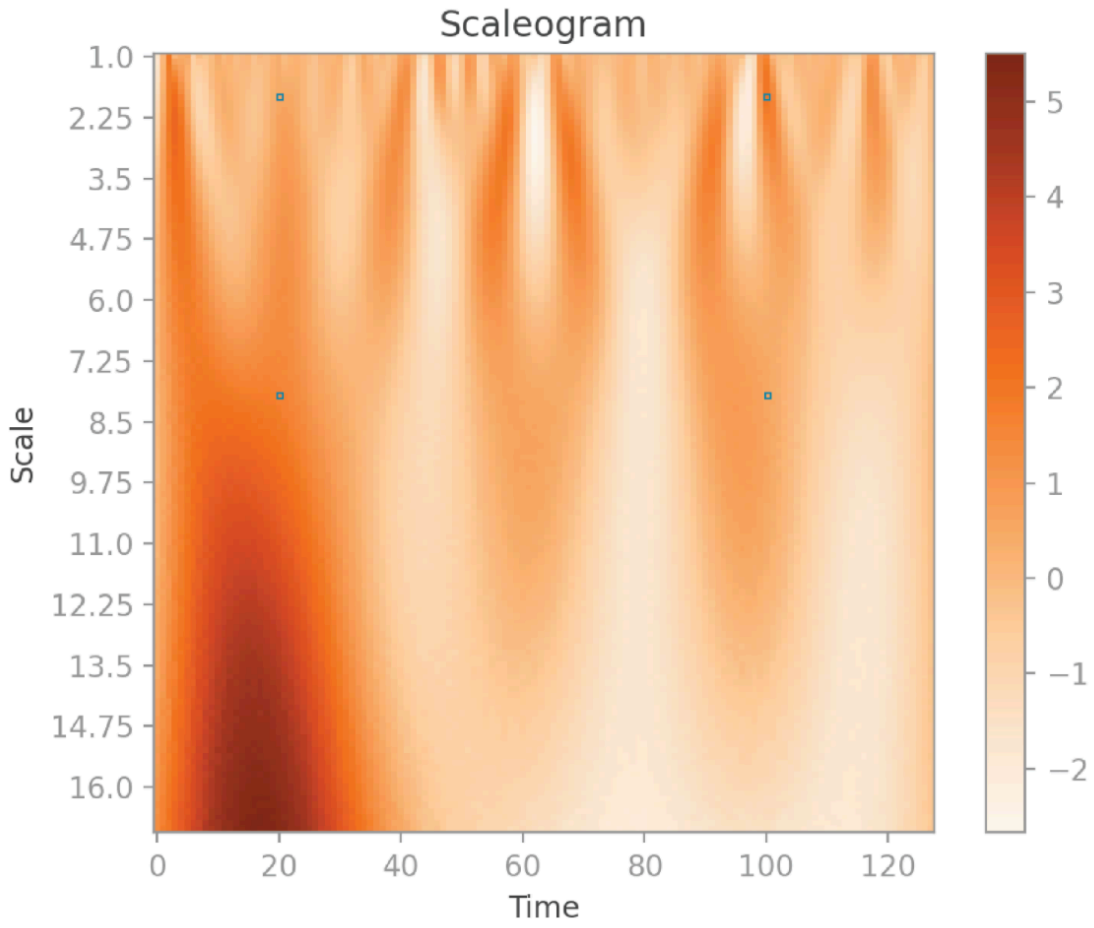

Рисунок 2.17 Скалеограма

На спектрограмі горизонтальна вісь відповідає часу, вертикальна - частоті, а колір - амплітуді. Скалеограма ж схожа на спектрограму, але вертикальна вісь відповідає масштабу вейвлету. Можна уявити, що параметр масштабу скалеограми відіграє таку ж роль, як і частота на спектрограмі, але з оберненою залежністю: чим вищий масштаб, тим нижча частота. Однак, оскільки вейвлети мають ширину, яка змінюється залежно від масштабу (на відміну від гауссів з фіксованою шириною у формулюванні спектрограми), скалеограми краще виявляють низькі частоти і краще локалізують високі частоти, ніж спектрограми.

Ось рівняння для неперервного вейвлет-перетворення:

$$\mathcal{W}(f(t)) = W(s,t) = \int_{-\infty}^{\infty} f(\tau)\,\bar{\psi}_{s,t}(\tau)\,d\tau \tag{2.11}$$

Де $\psi_{s,t}(\tau)$ це:

$$\psi_{s,t}(\tau) = \frac{1}{\sqrt{s}}\psi\left(\frac{\tau - t}{s}\right) \tag{2.12}$$

$\boldsymbol{\psi}$ тут називається "материнським вейвлетом" і може бути будь-якою вейвлет-функцією (наприклад, вейвлет мексиканського капелюха). У сценарії перетворення Габора фільтр лише транслюється, тому нам потрібен лише один параметр *t* для керування транслюванням. У сценарії вейвлет-перетворення нам потрібно два параметри: *s* для масштабування вейвлету і *t* для його трансляції.

Подібно до перетворення Габора, ми беремо комплексну спряжену функції $\psi_{s,t}$, що дає нам $\bar{\psi}_{s,t}$. Оскільки вейвлет, який ми використовуємо, має лише дійсні значення, це не змінює наші обчислення.

Потім ми множимо сигнал *f* на наш перетворений і масштабований вейвлет $\bar{\psi}_{s,t}$, і обчислюємо інтеграл за всіма значеннями часу $\boldsymbol{\tau}$, щоб отримати $W(s,t)$.

2.9 Висновки до розділу 2

У другому розділі мого дипломного дослідження, я розглянув математичні основи, які лягли в основу моєї роботи з біометричної автентифікації користувача смартфона за допомогою даних акселерометра.

Проаналізував основні архітектури нейронних мереж, включаючи рекурентні нейронні мережі (RNN), довготривалу короткочасну пам'ять (LSTM), та згорткові нейронні мережі (CNN), які часто використовуються для обробки зображень.

Розглянув декілька найпоширеніших оптимізаторів - алгоритмів, що використовуються для мінімізації функції помилки (функції втрат), їхнє математичне підгрунтя.

У розділі також провів детальний аналіз різних метрик якості для оцінки ефективності моделей машинного навчання під час експериментів.

Розібрався, як працюють необхідні для мене в цьому дослідженні методи трансформації часових рядів – методи Фур’є, Габора та вейвлет-перетворення. Навів наглядні приклади, математичне підгрунтя та результати роботи перелічених підходів.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Характеристики обладнання та вибір мови програмування та фреймворків

У даному розділі описано архітектуру системи та вибір мови програмування та фреймворків для реалізації класифікації користувачів на основі згенерованих скалеограм. Вибір цих інструментів був обґрунтований їх потужністю, широким спектром функціональності та зручністю використання.

1. Вибір мови програмування - Python: Мова програмування Python є однією з найпопулярніших мов у галузі дата саєнс та машинного навчання. Її простий та зрозумілий синтаксис робить Python дуже доступним для новачків і дозволяє розробникам швидко та ефективно створювати складні програми. Дата саєнс використовує Python як основну мову через його багатий екосистему бібліотек та фреймворків, спеціалізованих на обробці даних та машинному навчанні.

    - Бібліотека NumPy: NumPy є основною бібліотекою для обробки числових даних у Python. Вона надає підтримку для масивів та матриць, що дозволяє ефективно виконувати операції векторної та матричної математики. NumPy забезпечує швидке виконання обчислень завдяки використанню оптимізованого коду на мові C під капотом.

    - Бібліотека Pandas: Pandas є потужною бібліотекою для обробки та аналізу даних. Вона надає високопродуктивні та прості у використанні структури даних, такі як DataFrames, які дозволяють ефективно маніпулювати та аналізувати великі обсяги даних. Pandas також має розширені функції для обробки пропущених значень, групування даних, фільтрації, сортування та іншого.

- Бібліотека Matplotlib: Matplotlib є потужною бібліотекою для візуалізації даних у Python. Вона надає засоби для створення різноманітних графіків, діаграм, діагностичних зображень та іншого. Matplotlib дозволяє розробникам візуалізувати дані з високою точністю та налаштувати вигляд графіків згідно з вимогами проекту.

- Бібліотека scikit-learn: Scikit-learn є однією з найпопулярніших бібліотек для машинного навчання у Python. Вона надає широкий спектр алгоритмів для класифікації, регресії, кластеризації, виявлення аномалій та багатьох інших задач машинного навчання. Scikit-learn також має удобний та єдиностільний інтерфейс для побудови моделей, оцінки їхньої продуктивності та виконання операцій попередньої обробки даних.

- Бібліотека SciPy: SciPy є бібліотекою для наукових обчислень у Python. Вона надає численні алгоритми та інструменти для виконання різних обчислень, таких як оптимізація, інтерполяція, інтегрування, обробка сигналів, розв'язання диференціальних рівнянь та багато іншого. SciPy розширює можливості Python у галузі наукових та інженерних обчислень.

2. Вибір фреймворка PyTorch: Фреймворк PyTorch був обраний для реалізації нейронних мереж у даному дослідженні. PyTorch - це відкритий фреймворк для глибокого навчання, який зосереджений на гнучкості та простоті використання. Він надає високорівневі інтерфейси для побудови та тренування нейронних мереж, а також низькорівневі можливості для налаштування та оптимізації моделей. PyTorch використовує векторну математику та має оптимізовану обчислювальну базу на мові C++, що дозволяє прискорити виконання обчислень. Він також підтримує використання графічних процесорів (GPU) для прискорення тренування моделей, що є важливим фактором для розв'язання завдань машинного навчання, які вимагають великої обчислювальної потужності.

3. Технічні характеристики системи: Для візуалізації сирих даних, їхнього препроцессінгу та генерації скалеограм з часових рядів був використаний ноутбук з процесором Apple M1, 16 ГБ оперативної пам'яті (RAM). Ця конфігурація надає достатньо потужності для виконання такого роду завдань, але не годиться для ефективного й швидкого навчання різнотипних нейронних мереж, адже не має потужної відеокарти та графічного прискорювача.

4. Використання платформи Kaggle: Для навчання нейронних мереж, їх валідації та проведення досліджень загалом була використана платформа Kaggle. Kaggle - це сервіс для проведення досліджень у галузі машинного навчання та роботи з даними. Датасет зі згенерованими зображеннями-скалеограмами був завантажений саме на платформу Kaggle, що значно пришвидшило його обробку й завантаження до нейронних мереж. На Kaggle доступні потужні обчислювальні ресурси, зокрема майже безкоштовне використання графічних процесорів (GPU) (дозволяється використовувати графічні прискорювачі лише 30 годин на 1 тиждень). Для цього проекту було використано GPU Tesla P100 або Tesla GPU T4x2, що дозволило значно прискорити тренування моделей та обчислення метрик якості нейронних мереж.

5. У ході роботи було досліджено і створено нейронні мережі на базі кількох популярних архітектур згорткових нейронних мереж (CNN), включаючи ResNet, DenseNet, VGG та EfficientNet_B0. Кожна з цих архітектур має свої особливості та переваги.

    - ResNet (Residual Neural Network): ResNet є однією з найвідоміших та успішних архітектур CNN. Вона використовує концепцію "residual learning" для забезпечення глибоких мереж з легкістю навчання. ResNet має глибокі шляхи, які дозволяють пропускати деякі шари, що допомагає уникнути проблеми з вивченням глибоких мереж. Ця

архітектура відома своєю здатністю ефективно вирішувати завдання класифікації зображень.

- DenseNet (Densely Connected Convolutional Network): DenseNet є іншою потужною архітектурою CNN, яка відрізняється від інших своєю щільною з'єднаністю. У DenseNet кожен шар мережі сполучений з кожним наступним шаром, що сприяє збереженню інформації та покращує градієнтні потоки. Ця архітектура дозволяє зменшити кількість параметрів і зробити мережу більш компактною, що поліпшує її ефективність.

- VGG (Visual Geometry Group): VGG є однією з класичних архітектур CNN і відома своєю глибокою структурою зі значною кількістю згорткових шарів. Вона має просту та однорідну архітектуру зі згортковими шарами розміру 3x3 та пулінговими шарами розміру 2x2. VGG добре працює на широкому спектрі завдань класифікації зображень і вважається ефективною архітектурою.

- EfficientNet_B0: EfficientNet_B0 є однією з найновіших архітектур CNN, яка відзначається високою ефективністю та малою кількістю параметрів. Ця архітектура використовує метод автоматичного масштабування глибини, ширини та роздільної здатності мережі для досягнення оптимальної продуктивності. EfficientNet_B0 демонструє високу точність при класифікації зображень і відома своєю ефективністю на ресурсно обмежених пристроях.

3.2 Первинний аналіз даних

Необроблений навчальний набір даних має розміри у 29563983 рядків × 5 колонок, а також містить дані про загалом 387 унікальних користувачів (рис. 3.1, 3.2).

```
[6]: import matplotlib.pyplot as plt

print("Size of the raw dataframe:", df_main.shape)

# Assuming you have the data in a DataFrame named 'df'
user_counts = df_main['user_id'].value_counts()

# Plotting the histogram
plt.bar(user_counts.index, user_counts.values, color="green")
plt.xlabel('user_id')
plt.ylabel('Record Count')
plt.title('Number of Records per user_id')
plt.show()
```

```
Size of the raw dataframe: (29563983, 5)
```

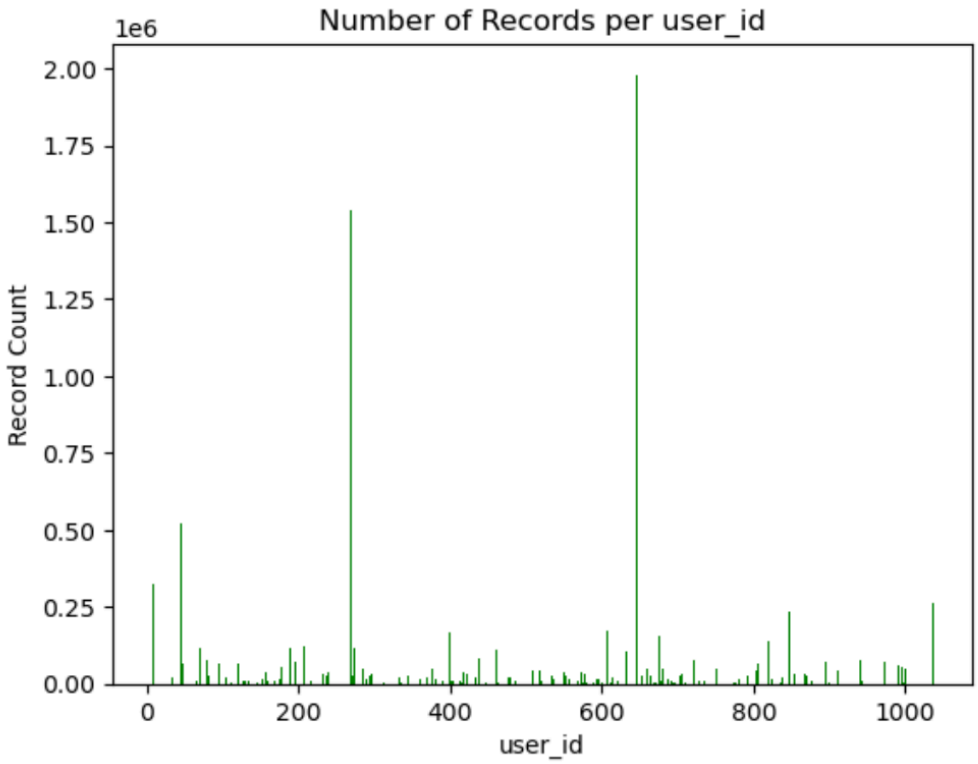

Рисунок 3.1 Кількість рядків даних на кожного користувача (у мільйонах)

```
[11]: df_raw
```

[11]:

	timestamp	x_axis	y_axis	z_axis	user_id
0	2012-05-10 10:17:48.311000064	0.340509	8.308413	4.140585	7
1	2012-05-10 10:17:48.531000064	0.381370	8.390134	4.249548	7
2	2012-05-10 10:17:48.753999872	0.272407	8.471856	4.018002	7
3	2012-05-10 10:17:48.971000064	0.149824	8.430995	4.290409	7
4	2012-05-10 10:17:49.191000064	0.272407	8.430995	4.481094	7
...	...	...	...	...	...
29563978	2012-05-13 06:56:10.584000000	-0.306000	-0.306000	-9.500000	1037
29563979	2012-05-13 06:56:10.660000000	-0.459000	-0.306000	-9.653000	1037
29563980	2012-05-13 06:56:10.828000000	-0.153000	-0.306000	-9.807000	1037
29563981	2012-05-13 06:56:10.843000064	-0.306000	-0.459000	-9.347000	1037
29563982	2012-05-13 06:56:10.843000064	-0.459000	-0.766000	-9.807000	1037

29563983 rows × 5 columns

Рисунок 3.2 Розміри необробленого навчального набору даних

Пропоную поглянути на візуалізацію акселерометричних даних трьох осей для конкретних користувачів у якийсь невеликий період часу, щоб порівняти ці дані (рис. 3.3, рис. 3.4).

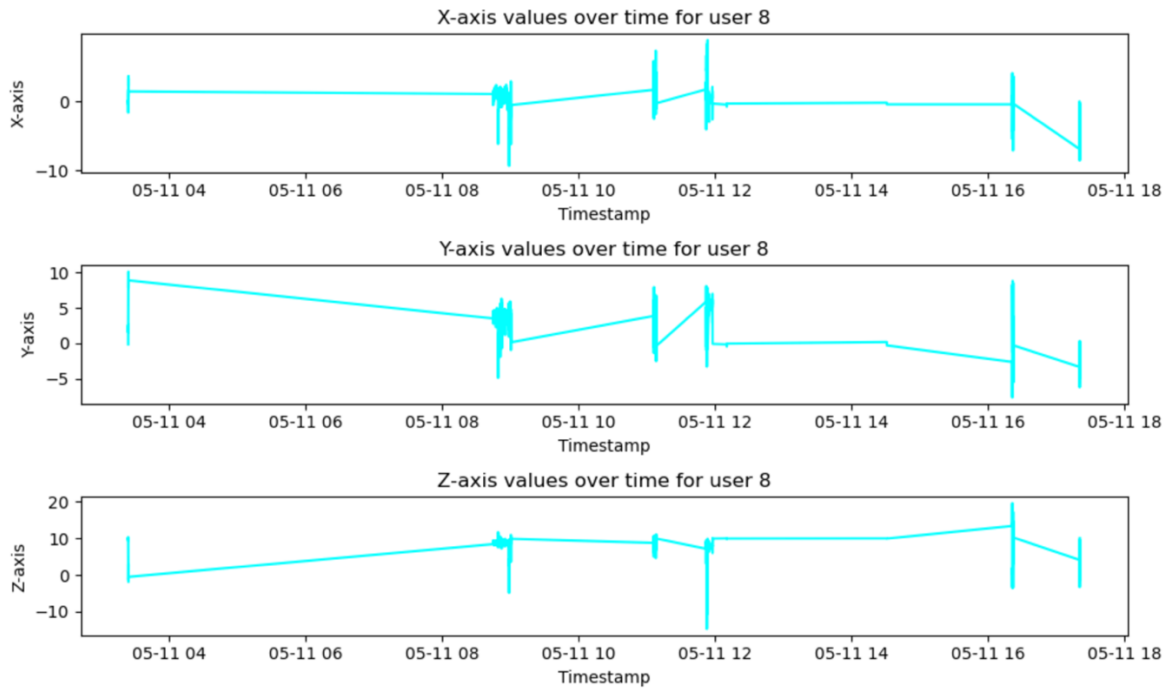

Рисунок 3.3 Акселерометричні дані для користувача під номером «8»

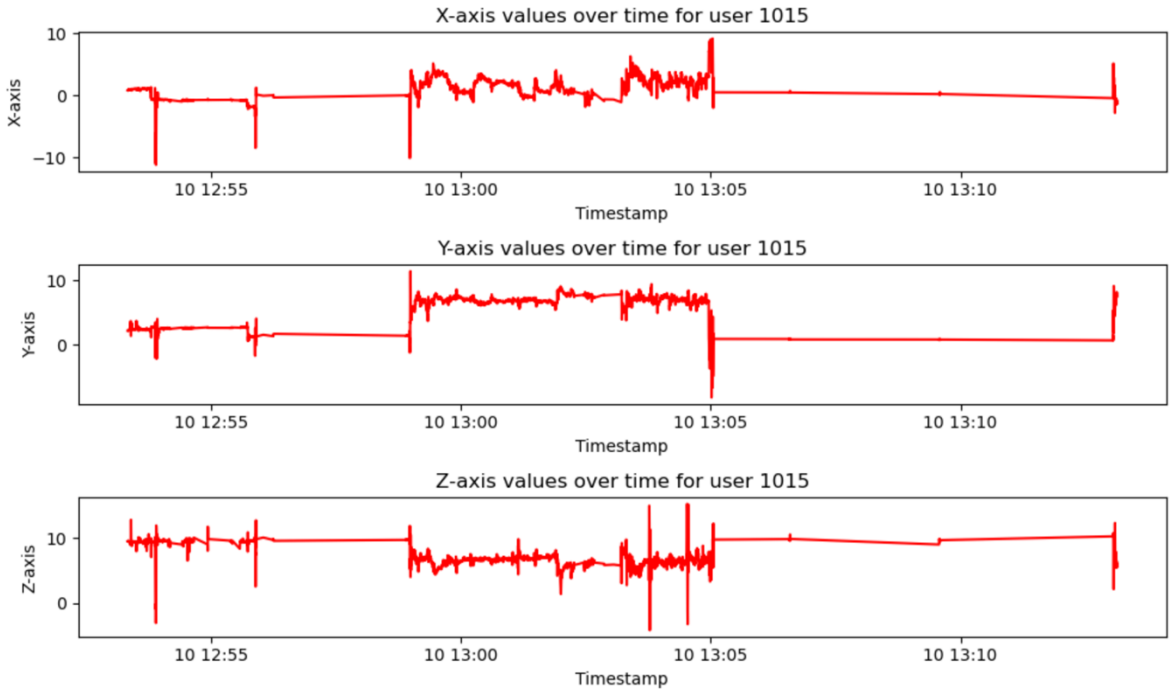

Рисунок 3.4 Акселерометричні дані для користувача під номером «1015»

Неозброєним оком видно, що дані сенсорів під час людської ходьби достатньо сильно відрізняються між собою у різних людей. Отже, машина (нейронна мережа) теж в перспективі може навчитись бачити в таких даних якісь шаблони, паттерни, правила і зможе ідентифікувати таким чином користувачів.

Але в своєму дослідженні я не буду працювати з «сирими» акселерометричними даними. Такі дослідження вже існують, і вони мають непогані результати, навіть використовуючи методи класичного машинного навчання по типу Support Vector Machine, KNN або Random Forest.

Я ж розробив нейронну мережу згорткового типу, яка ідентифікує користувачів за допомогою скалеограм – зображень, згенерованих прямо з акселерометричних даних (часових рядів). В наступному розділі я опишу процес конвертації часових рядів в скалеограми, які і будуть «паливом» для мого дослідження.

3.3 Створення скалеограм

3.3.1 Первинна обробка даних

Для того, щоб застосувати ці дані для навчання нейронних мереж, я їх обробив наступним чином.

По-перше, користувачі з недостатньою кількістю зразків виключаються з подальшого дослідження. Це пов'язано з тим, що методи глибокого навчання ефективно працюють з якомога більшою кількістю зразків. Тому я вирішив не розглядати тих користувачів для розробки моделей, у яких кількість вибірок менша за 30 000. Також було вирішено аналізувати та навчати нейронні мережі для різних підмножин користувачів (класах) – на 10 класах, на 25 та 50 класах. Таким чином я хотів порівняти ефективність нейронних мереж в різних умовах, за різної кількості цільових класів.

Отже, виокремивши випадковим чином 50 користувачів з-поміж 387, я зібрав рівно 30 000 рядків постійних (continuous) даних (наскільки це було можливо) для кожного з них, і поділив ці 30 000 рядків для кожного користувача (класа) на дві частини – 15 000 рядків відправились у навчальну вибірку, а інші 15 000 у тестову (рис. 3.5).

[14]:
```
import matplotlib.pyplot as plt


# Assuming you have the data in a DataFrame named 'df'
user_counts = df_30k['user_id'].value_counts()

# Plotting the histogram
plt.bar(user_counts.index, user_counts.values, color="r")
plt.xlabel('user_id')
plt.ylabel('Record Count')
plt.title('Number of Records per user_id')
plt.show()
```

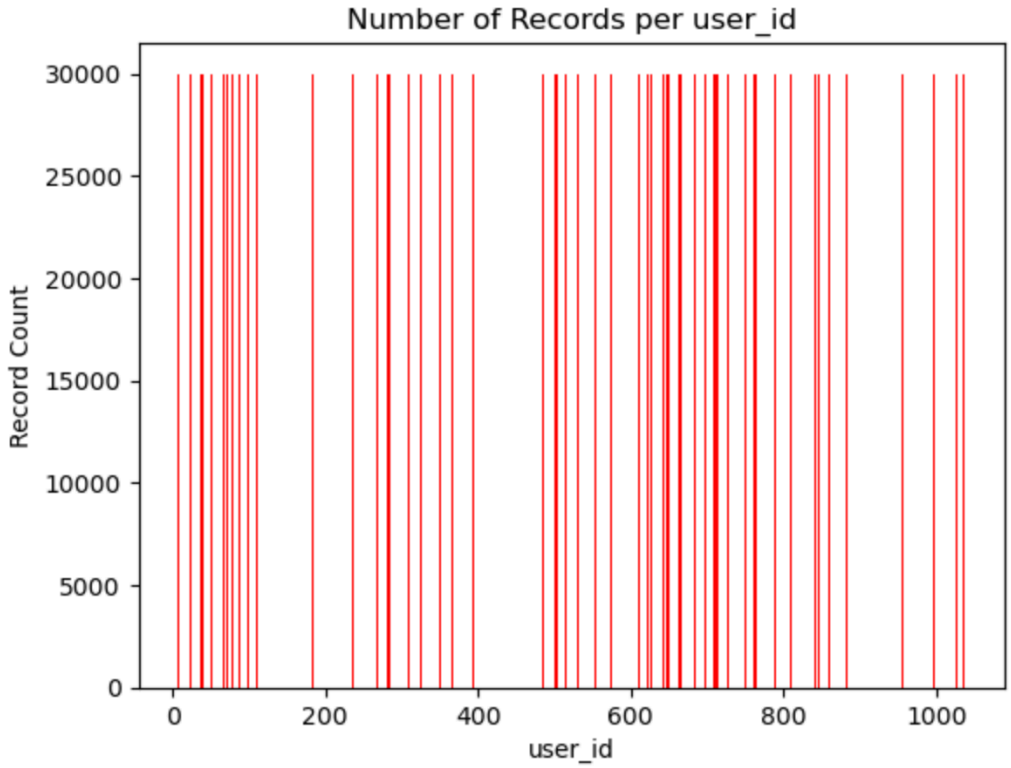

Рисунок 3.5 Результуючий датафрейм, точніше кількість рядків на одного користувача (всього 50 користувачів), який згодом буде поділено на дві частини порівну – навчальну і тестову вибірку даних.

Частота дискретизації даних акселерометра становить 5 Гц, що відповідає 5 вибіркам даних, які реєструються щосекунди. Отже, 15 000 вибірок даних відповідають 15 000/5 =3000 с = 50 хв. Іншими словами, мій набір навчальних даних містив 50 хвилин ходьби кожного користувача. Всього моделі я навчав на 10, 25 та 50 різних класах (користувачах), отже, в сумі я використовував підмножину з 750,000 рядків сирих акселерометричних даних. Вийшло 2 CSV файли розміром 47.5 Мб – навчальні і тестові дані.

Отже, сигнал даних акселерометра містить часові ряди даних за 3 осями (x, y та z). Часовий ряд даних кожного користувача розбивається на невеликі вікна, що складаються з 225 вибірок даних (тобто w=225), з урахуванням коефіцієнта зсуву (shift factor) *sf*=30. Іншими словами, якщо вікно містить вибірки з індексами від i до i+225, то наступне вікно міститиме вибірки з індексами від (i+30)+1 до i+256 . Таким чином, вікно значень для кожної осі акселерометра буде по суті 1D-масивом з 225 значень. Кожне вікно відповідає 45 с даних про ходьбу окремого користувача.

3.3.2 Вейвлет-перетворення

В цьому розділі я поясню, як я перетворив часові ряди акселерометричних даних у зображення – скалеограми. Повний код див. у Додатку А.

Починається код з імпорту необхідних бібліотек, таких як *pandas, pywt, numpy, matplotlib.pyplot, os* та *time*. Ці бібліотеки надають зручні функції для обробки даних, реалізації вейвлет-аналізу, маніпуляцій з масивами, візуалізації та роботи з операційною системою.

Основна частина коду розпочинається з циклу, який обробляє набори даних – спочатку *df_train*, а після нього й *df_test*.

Далі визначаються параметри вейвлету, які використовуються для аналізу. Я пробував створювати вейвлети за допомогою функції «Mexican hat» та «Morlet». Приклади створених зображень див. нижче.

Після цього встановлюються параметри для обробки даних, такі як розмір вікна *window_size* та крок *shift_factor*. Ці параметри визначають, які частини даних будуть розглядатись у кожній ітерації.

Наступний крок - обробка кожного користувача з набору даних. Для кожного користувача виконується наступна послідовність дій:

1. Отримуються дані, що відповідають поточному користувачу.
2. Перевіряється, чи існує папка для збереження результатів обробки поточного користувача. Якщо папка вже існує, обробка для цього користувача пропускається, інакше створюється нова папка для збереження результатів.
3. Для кожного вікна даних шукаються сигнали для трьох осей - *x_axis, y_axis* та *z_axis.*
4. З кожного сигналу застосовується функція *_create_scaleogram* для отримання скалеограми.
5. Отримані скалеограми для трьох осей конкатенуються в одну скалеограму *scaleogram_combined.*
6. Отримана скалеограма візуалізується за допомогою *matplotlib.pyplot* і зберігається у відповідну папку.
7. Процес повторюється для кожного вікна даних поточного користувача.

Після обробки всіх користувачів виводиться повідомлення про завершення обробки набору даних, а також час, який зайняло виконання коду для даного набору даних.

Цей код дозволяє ефективно перетворити часові ряди даних у скалеограми, які можуть бути використані для подальшого аналізу та розв'язання різних завдань, пов'язаних з обробкою сигналів.

Загалом було створено по 492 фотографії (scaleogram) для кожного користувача, тобто в сумі мій датасет містив 24600 зображень і 50 класів – а це 4 Гб даних для методу «mexh» та 6 Гб даних для методу «morl» (маю на увазі дані і навчальні, і тестові, в сумі).

На рис. 3.6, рис. 3.7, рис 3.8 та рис 3.9 можете побачити приклади згенерованих скалеограм (на вікнах даних) для трьох осей методом «mexh» за різних параметрів *scale*:

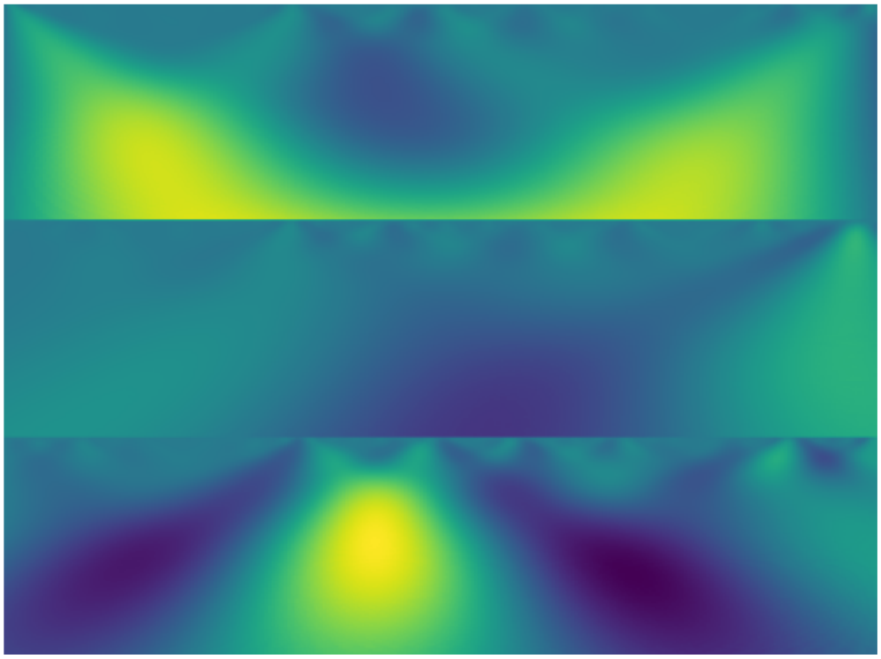

Рис. 3.6 Метод «mexh», *scale*=4

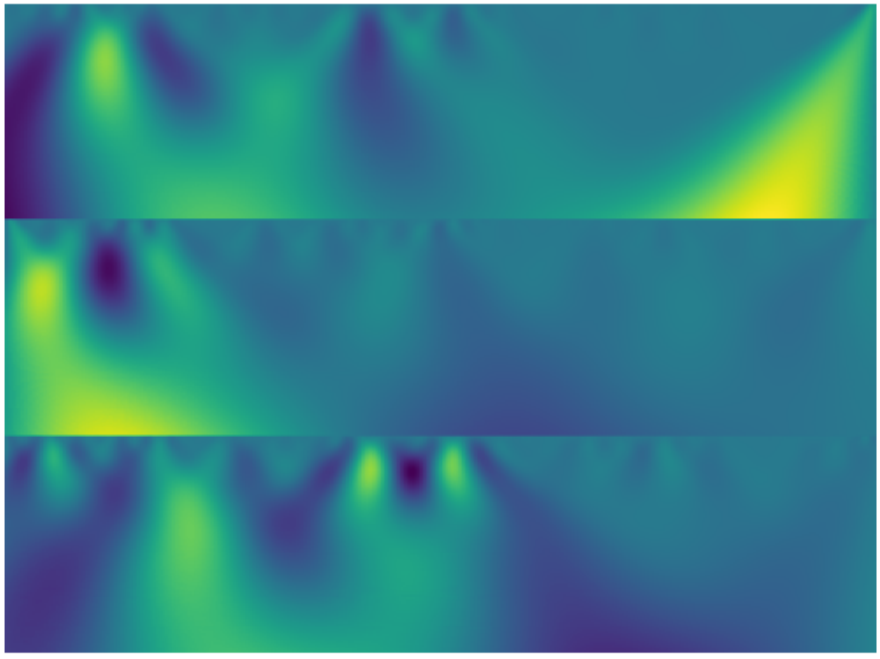

Рис. 3.7 Метод «mexh», *scale*=8

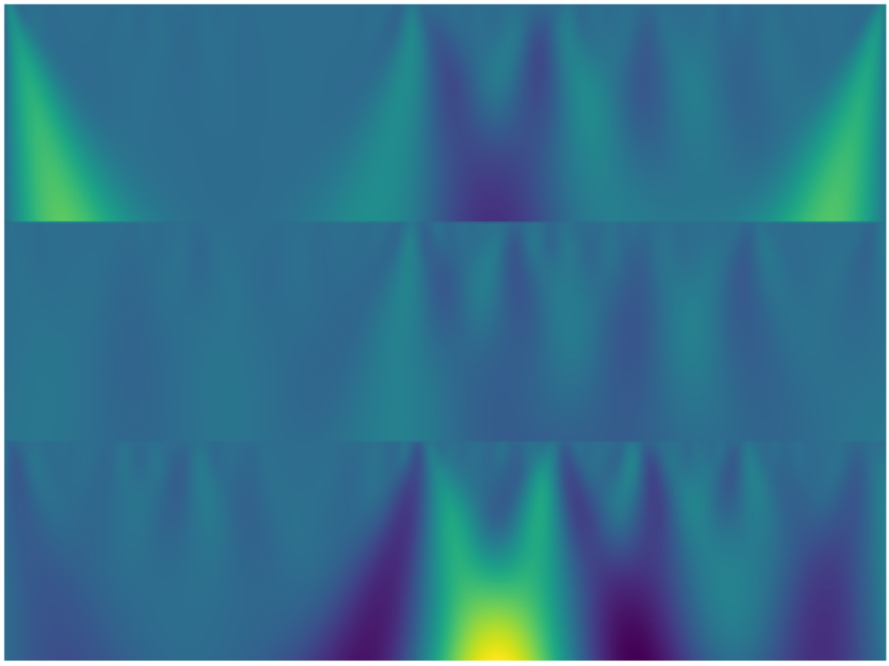

Рис. 3.8 Метод «mexh», *scale*=16

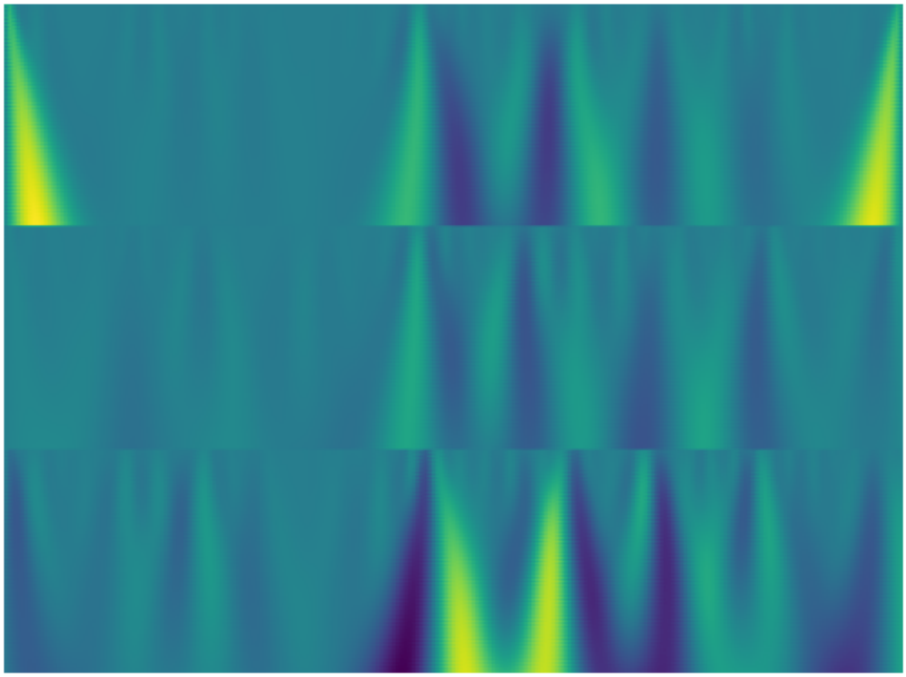

Рис. 3.9 Метод «mexh», *scale*=32

На рис. 3.10, рис. 3.11, рис 3.12 та рис 3.13 можете побачити приклади згенерованих скалеограм (на вікнах даних) для трьох осей методом «morl» за різних параметрів *scale*:

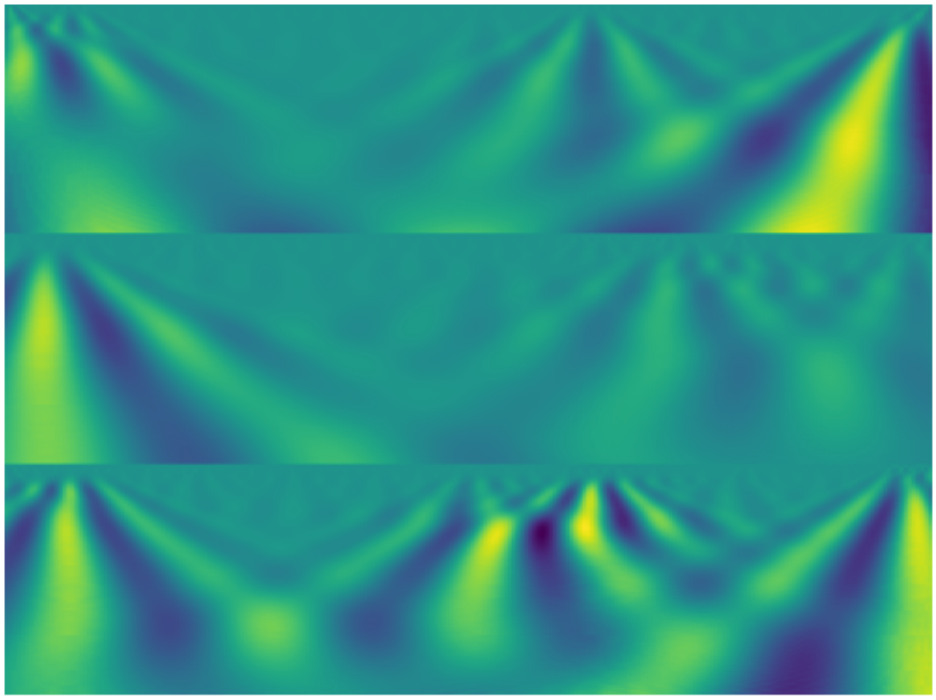

Рис. 3.10 Метод «morl», *scale*=4

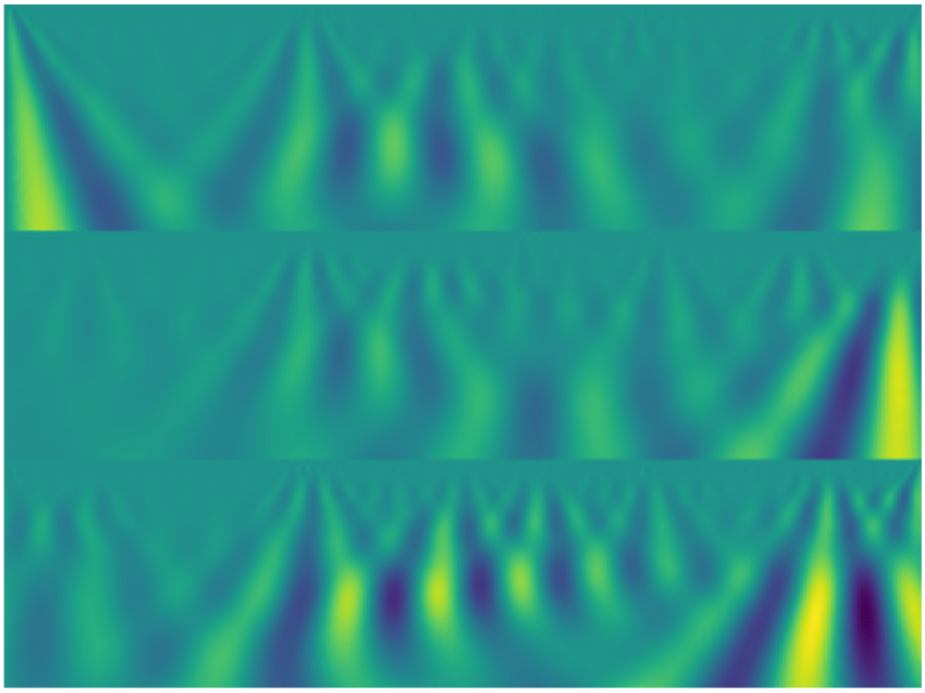

Рис. 3.11 Метод «morl», *scale*=8

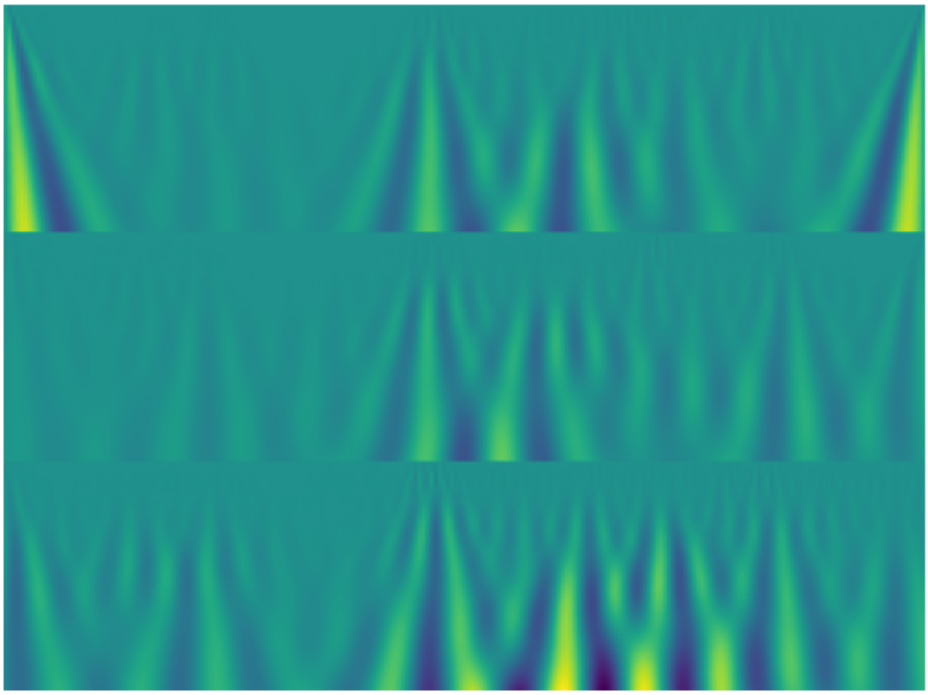

Рис. 3.12 Метод «morl», *scale*=16

Рис. 3.13 Метод «morl», *scale*=32

3.4 Порівняльний аналіз навчених моделей

Розгляньмо результати навчання таких нейронних мереж як ResNet18, ResNet34, DenseNet121, а також власної архітектури, а потім проаналізуймо порівняльну таблицю, щоб визначити найоптимальніше і найкраще рішення.

Я експериментував з різними вейвлет-перетвореннями та різною кількістю класів, як я вже казав в попередніх розділах. Моєю ціллю було визначити, які моделі загалом справляться краще з «полегшеною» задачею - класифікацією 10 класів, наприклад, а потім вже запустити ці ж моделі, але на більшому датасеті зображень з більшою кількістю класів.

Всі мої експерименти мають деякі спільні налаштування, такі як:

- Розмір батча – 32
- Функція витрат – Перехресна ентропія
- Оптимізатор – Adam
- Learning rate – 10^{-3}

Тому у розділі нижче я наводжу результати роботи різних моделей за різних умов, і визначаю найкращу. Також будую свою кастомну нейронну мережу, взявши до уваги результати роботи всіх попередніх.

3.4.1 Сімейство ResNet

- Архітектура – ResNet18
- Кількість епох – 40
- Кількість класів – 10
- Вейвлет – «mexh», *scale*=16

Рис. 3.14 Loss мережі типу ResNet18, задача класифікації 10 класів

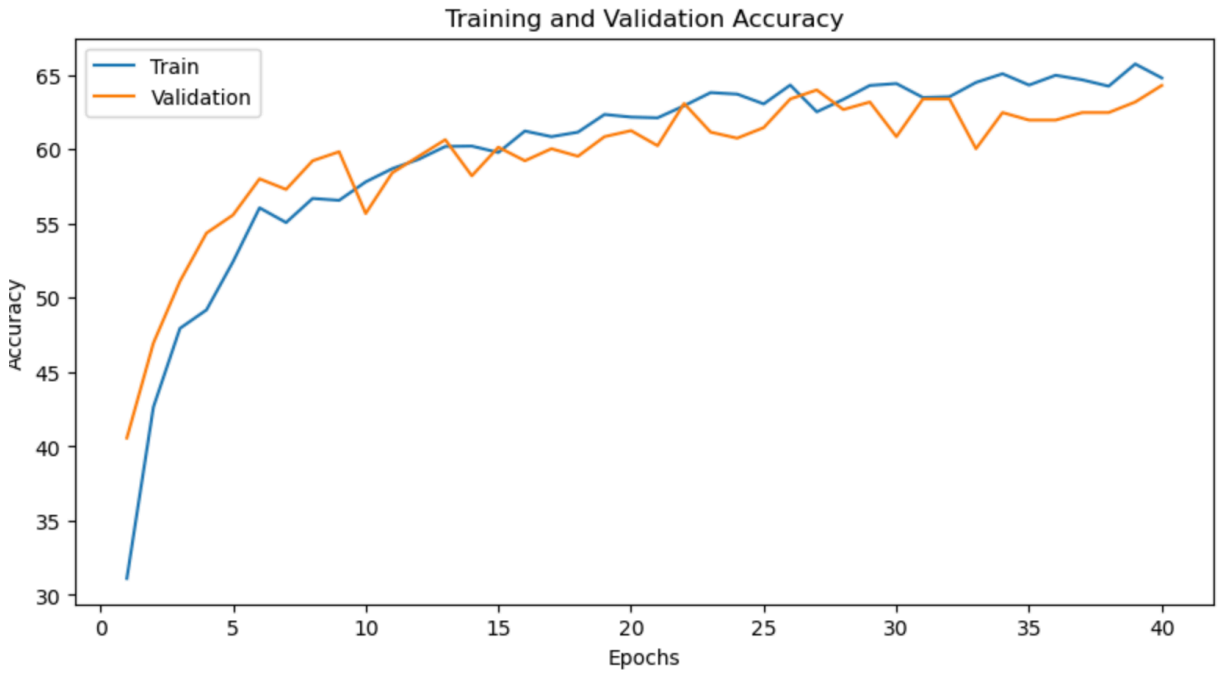

Рис. 3.15 Accuracy мережі типу ResNet18, задача класифікації 10 класів

Як видно із рисунків 3.14 та 3.15, модель доволі повільно вчиться вирішувати поставлену задачу, досягаючи свого максимуму у 65% точності приблизно на 15 епосі, після якої loss вже фактично перестає зменшуватись.

У таблиці 3.1 наведені точні значення accuracy та loss для тренувальних, валідаційних та тестових даних на кінець навчання моделі.

Результати навчання моделі на основі ResNet18

	Train	Validation	Test
Accuracy	0.6575	0.6318	0.6231
Loss	0.0631	0.0703	0.0733

Таблиця 3.1

- Архітектура – ResNet34
- Кількість епох – 20
- Кількість класів – 10
- Вейвлет – «mexh», *scale*=16

Рис. 3.16 Loss мережі типу ResNet34, задача класифікації 10 класів

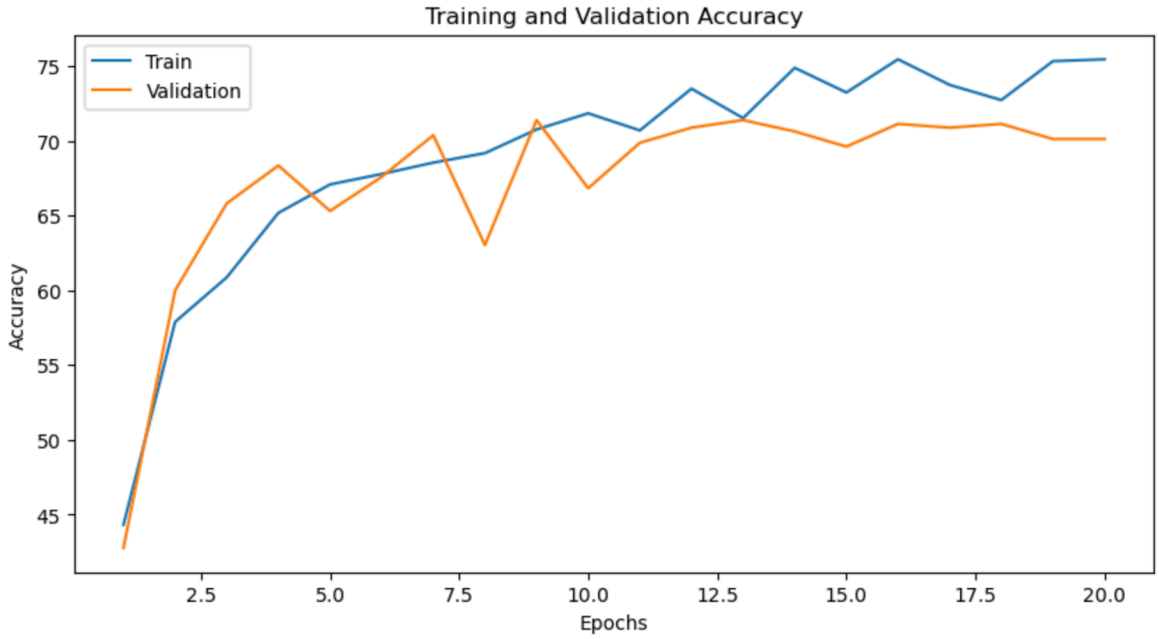

Рис. 3.17 Accuracy мережі типу ResNet34, задача класифікації 10 класів

Як видно із рисунків 3.16 та 3.17, ця модель швидко досягає свого максимального результату. Навіть швидше, ніж попередник. ResNet34 починає стагнацію навколо позначки в 70% точності після 10 епохи.

У таблиці 3.2 наведені точні значення accuracy та loss для тренувальних, валідаційних та тестових даних на кінець навчання моделі.

Результати навчання моделі на основі ResNet34

	Train	Validation	Test
Accuracy	0.7546	0.7113	0.6831
Loss	0.0419	0.0457	0.0483

Таблиця 3.2

3.4.2 Сімейство DenseNet

- Архітектура – DenseNet121
- Кількість епох – 35
- Кількість класів – 10
- Вейвлет – «mexh», *scale*=16

Рис. 3.18 Loss мережі типу DenseNet121, задача класифікації 10 класів

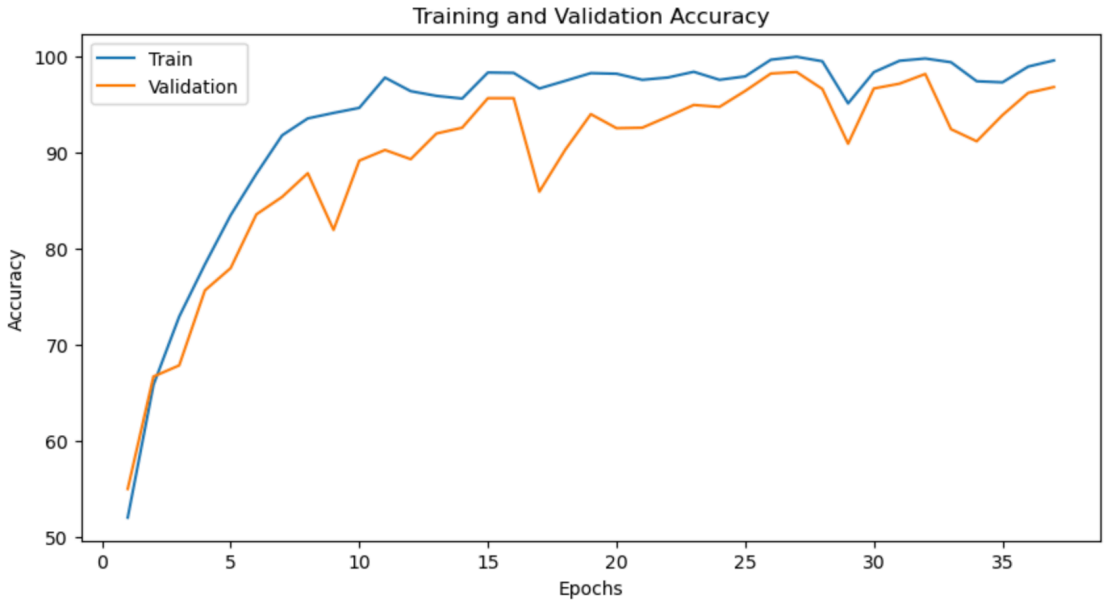

Рис. 3.19 Accuracy мережі типу DenseNet121, задача класифікації 10 класів

Як видно із рисунків 3.18 та 3.19, DenseNet показала себе сходу дуже добре, пробивши позначку у 80% точності уже на 7 епосі, а позначку у 90% на 11 епосі. Досяглу максимуму на останній епосі, показавши приголомшливий результат у 96.78% точності на валідації.

У таблиці 3.3 наведені точні значення accuracy та loss для тренувальних, валідаційних та тестових даних на кінець навчання моделі.

Результати навчання моделі на основі DenseNet121

	Train	Validation	Test
Accuracy	0.9955	0.9678	0.8544
Loss	0.0004	0.0033	0.0061

Таблиця 3.3

- Архітектура – DenseNet121
- Кількість епох – 40
- Кількість класів – 25
- Вейвлет – «morl», *scale*=8

Рис. 3.20 Loss мережі типу DenseNet121, задача класифікації 25 класів

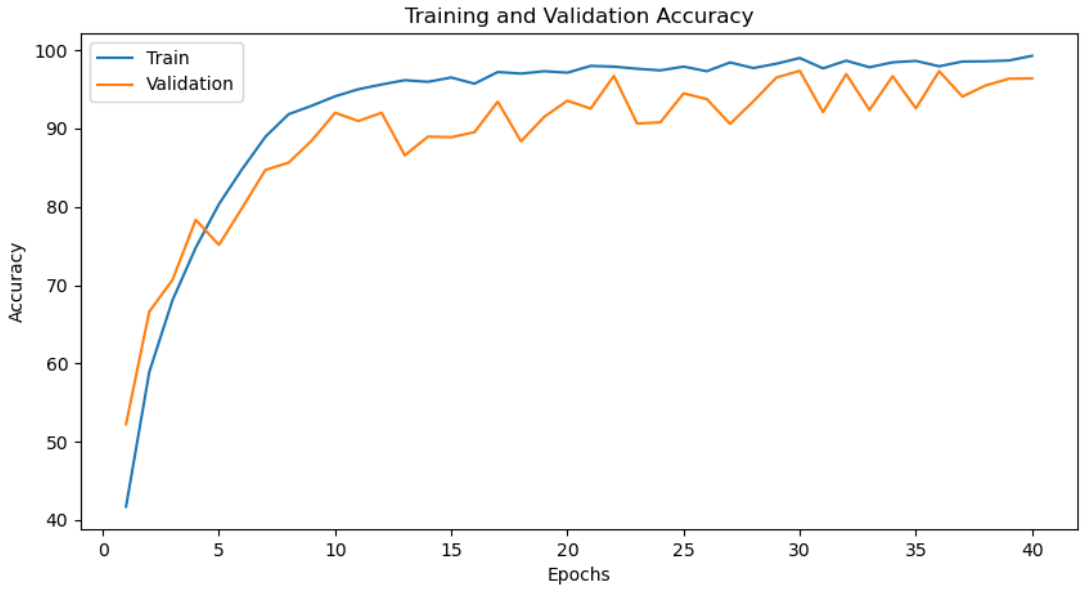

Рис. 3.21 Accuracy мережі типу DenseNet121, задача класифікації 25 класів

Як видно із рисунків 3.20 та 3.21, DenseNet навіть на 25 класах і з інакшим вейвлет-перетворенням показала себе теж дуже добре, пробивши позначку у 90% на 10 епосі. Але зрештою, почала стагнацію до самого кінця навчання і в основному знаходилась в межах 90-95% точності, що все рівно є дуже хорошим результатом.

У таблиці 3.4 наведені точні значення accuracy та loss для тренувальних, валідаційних та тестових даних на кінець навчання моделі.

Результати навчання моделі на основі DenseNet121

	Train	Validation	Test
Accuracy	0.9867	0.9423	0.8823
Loss	0.0012	0.0018	0.0057

Таблиця 3.4

- Архітектура – DenseNet121
- Кількість епох – 40
- Кількість класів – 50
- Вейвлет – «mexh», *scale*=16

Рис. 3.22 Loss мережі типу DenseNet121, задача класифікації 50 класів

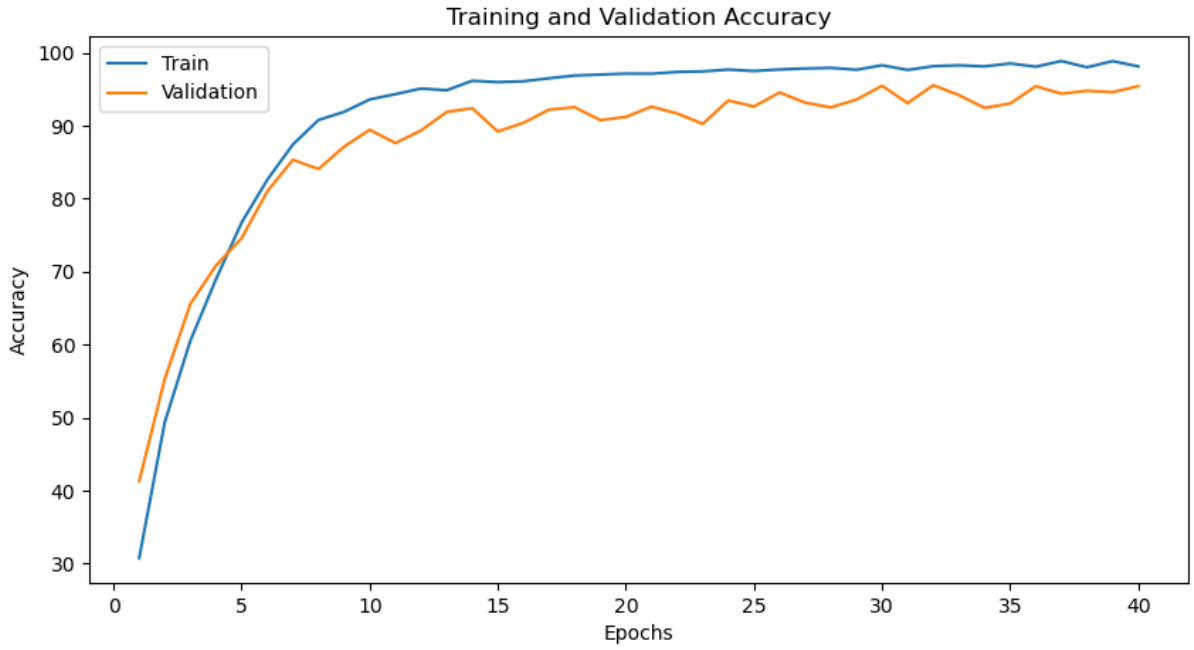

Рис. 3.23 Accuracy мережі типу DenseNet121, задача класифікації 50 класів

Як видно із рисунків 3.22 та 3.23, DenseNet на фото, створених за допомогою вейвлета «mexh» та *scale*=16 на задачі з 50 класами (максимальна кількість даних – 24600 фото) показав дуже хороші результати.

Точність на рівні 92%, починаючи з 12 епохи, але loss перестала зменшуватись після 10 епохи. Тому тут ми бачимо невелике перенавчання.

У таблиці 3.5 наведені точні значення accuracy та loss для тренувальних, валідаційних та тестових даних на кінець навчання моделі.

Результати навчання моделі на основі DenseNet121

	Train	Validation	Test
Accuracy	0.9885	0.9541	0.8812
Loss	0.0021	0.0010	0.0037

Таблиця 3.5

3.4.3 Запропонована архітектура

Після багатьох експериментів з даними, архітектурами й гіперпараметрами, я зумів створити нейронну мережу, що показала кращі результати, ніж всі попередні моделі.

Моя модель базується на архітектурі сімейства EfficientNet_B0.

Такі мережі вирізняються своєю ефективністю та високою точністю в завданнях класифікації зображень. Вони були розроблені з метою досягнення кращих результатів при мінімальному споживанні ресурсів.

Сама EfficientNet_B0 має дуже ефективну архітектуру, яка базується на компаунд масштабуванні. Вона складається з послідовної комбінації трьох основних компонентів: згорткових шарів (Convolutional layers), блоків мобільних інвертованих згорток (Mobile Inverted Residual blocks) та блоків перетворень (Transform blocks).

Основною ідеєю, яка лежить в основі EfficientNet, є збільшення глибини, ширини та роздільної здатності моделі, а також оптимізація кожного компонента для досягнення найкращого співвідношення між точністю та обчислювальними витратами.

Архітектура EfficientNet_B0 має 7 етапів (blocks), які формують глибоку нейронну мережу. Кожен етап складається з мобільних інвертованих згорток, які включають конволюційні шари, функції активації та згорткові блоки для зменшення розмірності.

Результуюча модель має близько 5.3 мільйона параметрів. Це включає ваги для конволюційних шарів, мобільних інвертованих згорток та блоків перетворень.

Відносно кількості шарів, то вона має загальну кількість 208 шарів.

Отже, результати:

- Архітектура – На основі EfficientNet_B0
- Розмір батча – 32
- Функція витрат – Перехресна ентропія
- Оптимізатор – Adam
- Learning rate – 10^{-3}
- Кількість епох – 25
- Кількість класів – 50
- Вейвлет – «mexh», *scale*=16

Рис. 3.24 Loss запропонованої мережі, задача класифікації 50 класів

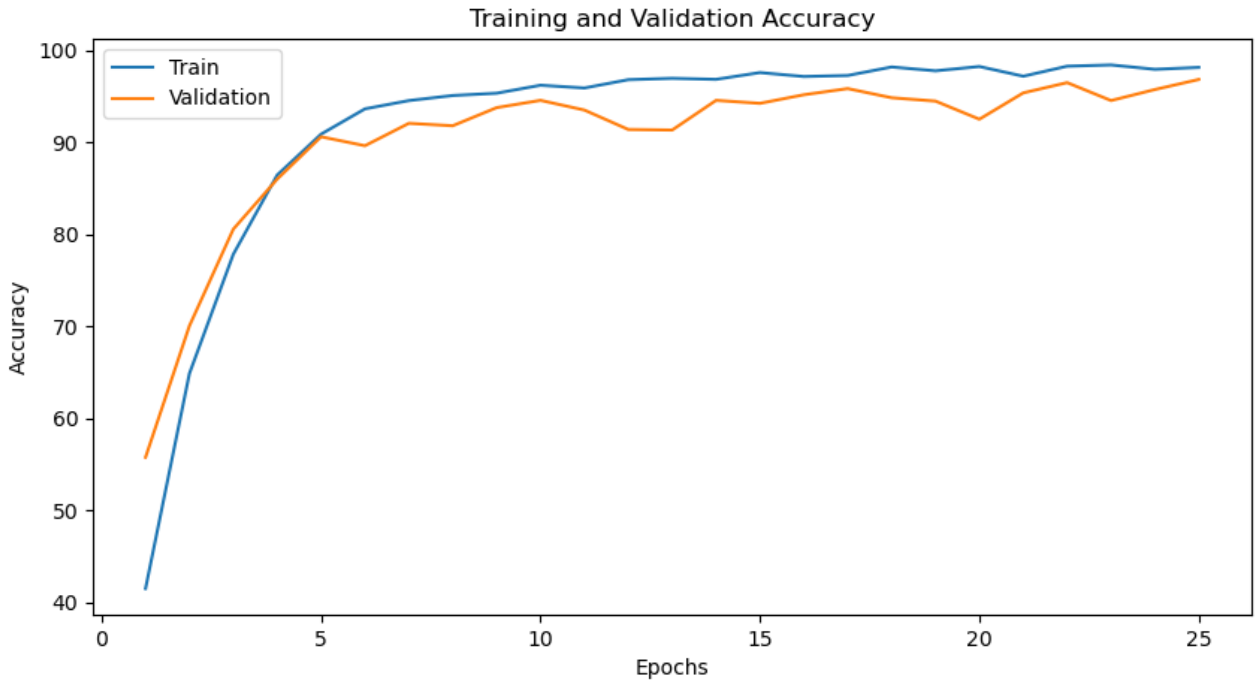

Рис. 3.26 Accuracy запропонованої мережі, задача класифікації 50 класів

Як видно з графіку точності, accuracy піднялась вище за 90% уже на 5 епосі! З того часу валідаційна точність повільно але впевнено повзла догори, подолавши поріг у 95% приблизно на 23 епосі.

Loss цієї моделі впродовж навчання опустився до позначки у 0.01, що є хорошим результатом. Під кінець навчання він досягнув таки відмітки у 0.005.

У таблиці 3.6 наведені точні значення accuracy та loss для тренувальних, валідаційних та тестових даних на кінець навчання моделі.

Результати навчання запропонованої моделі

	Train	Validation	Test
Accuracy	0.9814	0.9683	0.9253
Loss	0.0016	0.0036	0.0052

Таблиця 3.6

3.5 Висновки до розділу 3

Порівняння результатів роботи різних моделей

Архітектура	Кількість класів	Accuracy			Loss		
		train	validation	test	train	validation	test
ResNet18	10	0.6575	0.6318	0.6231	0.0631	0.0703	0.0733
ResNet34	10	0.7546	0.7113	0.6831	0.0419	0.0457	0.0483
DenseNet121	10	0.9955	0.9678	0.8544	0.0004	0.0033	0.0061
DenseNet121	25	0.9867	0.9423	0.8823	0.0012	0.0018	0.0057
DenseNet121	50	0.9885	0.9541	0.8812	0.0012	0.0018	0.0057
Власна модель	50	0.9814	0.9683	0.9253	0.0016	0.0036	0.0052

Таблиця 3.7

Блакитним кольором я виділив показники найкращої моделі.

Як видно із таблиці 3.7, запропонована мною модель, навчена на даних всіх 50 користувачів, а це 24600 зображень-скалеограм, виявилася найкращою за основними метриками якості. Вона заснована на EfficientNet_B0, що стало потужним підгрунтям її успіху. Загалом можна виокремити наступні моменти, завдяки яким модель на основі EfficientNet-B0 переважає інші розглянуті моделі:

1. Краща ефективність: така модель використовує метод компаунд-скейлінгу, що дозволяє автоматично змінювати розмір моделі залежно від об'єму даних та обчислювальних ресурсів. Це дозволяє досягати кращої ефективності і точності, використовуючи менше ресурсів.
2. Менша кількість параметрів: EfficientNet-B0 має меншу кількість параметрів порівняно з деякими іншими моделями, такими як ResNet і DenseNet. Це зменшує обчислювальну складність та сприяє швидшому тренуванню та використанню моделі.
3. Збереження ресурсів: EfficientNet-B0 використовує спеціальні типи шарів, такі як MBConv, які ефективно використовують обчислювальні ресурси.

Це дозволяє досягати високої точності з меншим споживанням енергії та часу.

4. Загальна архітектура: EfficientNet-B0 поєднує різні техніки, такі як глибокі згорткові шари, пакетна нормалізація, функції активації і згорткові блоки, що сприяють досягненню високої точності в класифікації зображень та інших завданнях обробки зображень.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінювання характеристик майбутнього програмного продукту, спеціалізованого на дослідженні демографічного стану. Реалізація цього продукту дозволить проводити необхідні дослідження, що впливатимуть на якість досліджень не лише в Україні, але й у всьому світі. Також представлено різні варіанти реалізації, які сприятимуть вибору оптимальної стратегії з урахуванням економічних факторів та сумісності з програмним продуктом. Для цього використовується функціонально-вартісний аналіз (ФВА), який оцінює реальну вартість продукту та допомагає знижувати витрати шляхом виявлення ефективніших варіантів виробництва.

4.1 Постановка задачі проєктування

У даній роботі використовується метод ФВА для проведення техніко-економічного аналізу системи біометричної аутентифікації користувачів смарфтону за допомогою даних акселерометра. З метою задоволення потреб кожної окремої підсистеми, рішення щодо проектування та реалізації компонентів повинні враховувати вплив на всю систему.

Технічні вимоги до представленого програмного продукту приведені нижче:

- функціонування на персональних комп’ютерах із стандартним набором компонентів;
- запуск і робота програмного забезпечення на базі смартфонів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;

- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який вирішує задачу біометричної автентифікації користувачів за допомогою показників акселерометра. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – вибір фреймворку машинного навчання.

F_3 – вибір середовища програмування.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python.

б) C++.

Функція F_2.

а) Pytorch.

б)TensorFlow.

Функція F_3:

а) Jupyter Notebook (Kaggle / Google Colab).

б) PyCharm.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

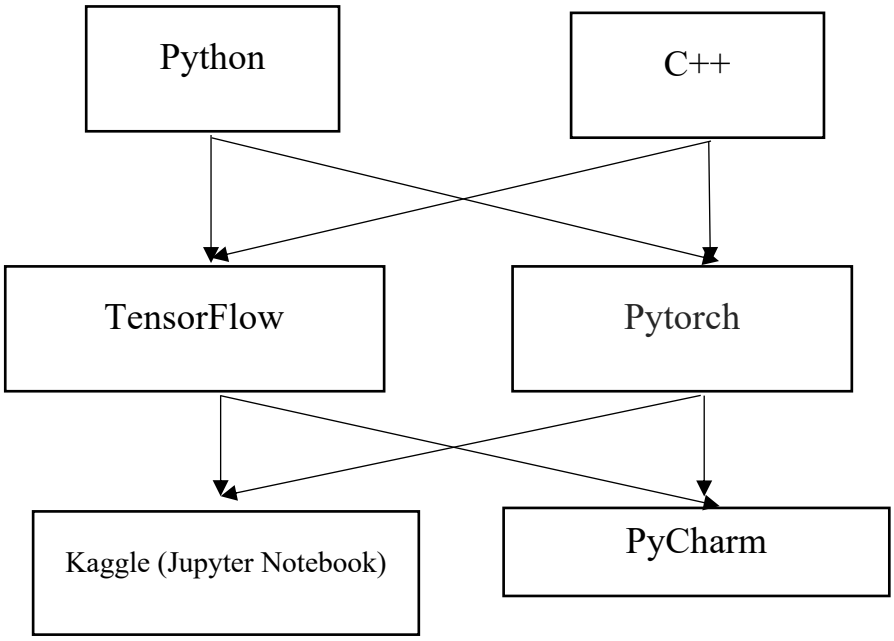

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Зручність, багатофункціональність, широкий вибір потужних бібліотек	Відносно низька швидкодія, обмежена підтримка апаратного прискорення
	Б	Висока швидкодія, доступ до оптимізаційних можливостей процесора,	Складність, менша кількість предметних бібліотек, вимога до вміння ручного керування пам'яттю
F_2	*А*	Широкий вибір моделей, розширені можливості управління пам'яттю	Складніший синтаксис, більша кількість коду для досягнення тих самих результатів

	Б	Простий синтаксис, динамічні графи, легкість у розробці та налагодженні моделей,	Менша швидкодія порівняно з TensorFlow на деяких завданнях
F_3	*А*	Інтерактивність, можливість візуалізації даних та результатів, зручність	Потребує багато ресурсів, обмежена підтримка великих обсягів даних,
	Б	Підтримка автодоповнення, інструменти для оптимізації, робота з великими проектами	Висока ціна платної версії, проблеми з налаштуванням деяких бібліотек

Таблиця 4.1.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1*:*

Перевагу даємо зручності, багатофункціональності, широкому вибору бібліотек. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2*:*

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3*:*

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

$F_1\text{а} - F_2\text{а} - F_3\text{а}$

F_1а – F_2б – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На підставі розглянутих вище даних визначаються основні параметри для вибору та розрахунку коефіцієнта технічного рівня програмного продукту. Для оцінки характеристик програмного продукту будуть використовуватись наступні параметри: швидкодія мови програмування (X1), об'єм пам'яті для обчислень та збереження даних (X2), час попередньої обробки даних (X3) і потенційний об'єм програмного коду (X4). Значення цих параметрів (гірші, середні і кращі) обираються відповідно до вимог замовника та умов експлуатації програмного продукту, як це показано у таблиці 4.2.

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	8000	12000	16000
Об’єм пам’яті	X2	Мб	128	64	32
Час попередньої обробки даних	X3	мс	3	2	1

Потенційний об’єм програмного коду	X4	кількість рядків коду	2000	1500	1000

Таблиця 4.2

4.4 Аналіз експертного оцінювання параметрів

Після докладного обговорення і аналізу кожен експерт оцінює важливість кожного параметра для конкретної цілі - розробки програмного продукту, який забезпечує точність результатів при обчисленні прогнозних значень та параметрів моделей адаптивного прогнозування. Значимість кожного параметра визначається методом попарного порівняння, за допомогою експертної комісії, що складається з 7 осіб.

Процес визначення коефіцієнтів значущості включає наступні кроки: встановлення рівня значимості параметра шляхом присвоєння рангів, перевірку придатності експертних оцінок, визначення оцінки попарного пріоритету параметрів, обробку результатів та визначення коефіцієнта значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Позначення параметра	Назва параметрів	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			

X1	Швидкодія мови програмування	оп/мс	1	1	2	2	1	1	2	10	-7.5	56.25
X2	Об’єм пам’яті	Мб	3	4	3	3	3	4	4	24	6.5	42.25
X3	Час попередньої обробки даних	мс	2	2	1	1	2	2	1	11	-6.5	42.25
X4	Потенційний об’єм програмного коду	кількість рядків коду	4	3	4	4	4	3	3	25	7.5	56.25
	Разом		10	10	10	10	10	10	10	70	0	197

Таблиця 4.3

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів,

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \tag{4.3}$$

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197. \tag{4.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2 \cdot (4^3-4)} = 0.754 > W_k = 0.67 \tag{4.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 4.4.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0.5
X1 і X3	<	<	>	>	<	<	>	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	>	>	>	>	>	>	>	>	1.5
X2 і X4	<	>	<	<	<	>	>	<	0.5
X3 і X4	<	<	<	<	<	<	<	<	0.5

Таблиця 4.4 Попарне порівняння параметрів

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A=\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Параметри	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	$b_i{}^1$	$K_{\text{ві}}{}^1$	$b^2{}_i$	$K_{\text{ві}}{}^2$
X1	1	0.5	0.5	0.5	2.5	0.16	9.25	0.16	34.125	0.16
X2	1.5	1	1.5	0.5	4.5	0.28	16.25	0.28	59.125	0.28
X3	1.5	0.5	1	0.5	3.5	0.22	12.25	0.21	41.875	0.2
X4	1.5	1.5	1.5	1	5.5	0.34	21.25	0.35	77.875	0.36
Всього					16	1	59	1	213	1

Таблиця 4.5 Розрахунок вагомості параметрів

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об’єм пам’яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	Б	X1	100	25	0.16	4
F3	А	X2	87	29	0.28	8.12
	Б	X3	27	19	0.2	3.8
F4	А	X4	25	23	0.36	8.28

Таблиця 4.6 Розрахунок показників рівня якості варіантів реалізації основних функцій

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4 + 8.12 + 8.28 = 20.4;$$

$$K_{K2} = 4 + 3.8 + 8.28 = 16.08.$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Спочатку проводиться розрахунок трудомісткості для визначення вартості розробки програмного продукту. Всі варіанти розробки включають два окремі завдання: розробку проекту програмного продукту і розробку програмної оболонки.

Завдання 1, пов'язане з розробкою проекту програмного продукту, відноситься до групи А за ступенем новизни. Завдання 2, пов'язане з розробкою програмної оболонки, відноситься до групи Б. За складністю алгоритми, що використовуються в завданні 1, належать до групи 1, а в завданні 2 - до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 базується на використанні інформації у вигляді даних. Далі проводиться розрахунок норм часу на розробку та програмування для кожного з цих завдань.

Загальна трудомісткість обчислюється як:

$$\mathrm{Т_О} = \mathrm{Т_Р} \cdot \mathrm{К_П} \cdot \mathrm{К_{СК}} \cdot \mathrm{К_М} \cdot \mathrm{К_{СТ}} \cdot \mathrm{К_{СТ.М}}, \qquad (4.13)$$

де $\mathrm{Т_Р}$ – трудомісткість розробки ПП;

$\mathrm{К_П}$ – поправочний коефіцієнт;

$\mathrm{К_{СК}}$ – коефіцієнт на складність вхідної інформації;

$\mathrm{К_М}$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =31 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.6. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 31 \cdot 1.6 \cdot 0.8 = 39{,}68 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =25 людино-днів, $К_П$ =1.2, $К_{СК}$ = 1, $К_{СТ}$ =0.9:

$$Т_2 = 25 \cdot 1.2 \cdot 0.9 = 27 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (39{,}68 + 27 + 5.4 + 27) \cdot 8 = 792{,}64 \text{ людино-годин.}$$

$$Т_{II} = (39{,}68 + 27 + 7.22 + 27) \cdot 8 = 807{,}2 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 12345 грн., один аналітик в області даних з окладом 13456 грн. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.,} \qquad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$СЧ = \frac{12345 + 12345 + 13456}{3 \cdot 21 \cdot 8} = 75.68 \text{ грн.} \quad (4.15)$$

Тоді, розраховуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \quad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 75.68 \cdot 792,64 \cdot 1.2 = 71984.39$ грн.

II. $С_{ЗП} = 75.68 \cdot 807,2 \cdot 1.2 = 73306.67$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 71984.39 \cdot 0.22 = 15836.56$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 73306.67 \cdot 0.22 = 16127.46$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Оскільки одна ЕОМ обслуговує одного програміста з окладом 12345 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 12345 \cdot 0,2 = 29628 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 29628 \cdot (1 + 0.2) = 35553.6 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 35553.6 \cdot 0.22 = 7821.79 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 30000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.2 \cdot 0.25 \cdot 32000 = 9000 \quad грн.,$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.2 \cdot 30000 \cdot 0.05 = 1800 \quad грн.,$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 11) \cdot 8 \cdot 0.7 = 1355.2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1355.2 \cdot 0,5 \cdot 0,2 \cdot 4,86 = 658,62 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = = 30000 \cdot 0,67 = 20100 \quad грн.$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$С_{ЕКС} = 35553.6 + 7821.79 + 9000 + 1800 + 658.62 + 20100 = 74934.01$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 74934.01 / 1355.2 = 55.29 \quad грн/год.$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т, \qquad (4.18)$$

I. $С_М = 55.29 \cdot 792,64 = 43825.06$ грн.

II. $С_М = 55.29 \cdot 807,2 = 44630.08$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67, \qquad (4.19)$$

I. $С_Н = 71984.39 \cdot 0.67 = 48229.54$ грн.

II. $С_Н = 73306.67 \cdot 0,67 = 49115.46$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н, \qquad (4.20)$$

I. $С_{ПП} = 71984.39 + 15836.56 + 43825.06 + 48229.54 = 179875.55$ грн.

II. $С_{ПП} = 73306.67 + 16127.46 + 44630.08 + 49115.46 = 183179.67$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 20.4 / 179875.55 = 11.3411 \cdot 10^{-5},$$

$$К_{ТЕР2} = 16.08 / 183179.67 = 8.7782 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 11.3411 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було встановлено, що між двох проаналізованих варіантів, що залишились після першого відбору, найоптимальнішим для розробки цільового програмного забезпечення є перший варіант реалізації програмного продукту:

- Python;
- PyTorch;
- Jupyter Notebook (Kaggle / Google Colab).

У такої комбінації технологій виявився найкращий показник техніко-економічного рівня якості.

4.8 Висновки до розділу 4

Була поставлена задача проєктування програмного продукту, де були визначені цілі та область його застосування. Обґрунтовані функції програмного продукту, визначено необхідні можливості та сервіси для користувачів. Виконано обґрунтування системи параметрів програмного продукту, включаючи ключові налаштування та вимоги до продукту.

Проведено аналіз експертного оцінювання параметрів, де експерти оцінили важливість окремих параметрів та їх вплив на функціональність продукту. Здійснено аналіз рівня якості варіантів реалізації функцій, порівняно різні варіанти реалізації та оцінено їх ефективність. Проведено економічний аналіз варіантів розробки програмного продукту, включаючи оцінку фінансових витрат та прибутковості, здійснено вибір кращого варіанту програмного продукту на основі техніко-економічного аналізу

ВИСНОВКИ

В даній роботі було проведено широке дослідження використання даних акселерометра для біометричної автентифікації користувача смартфона. Аналітична робота, проведена в розділі 1, підтвердила актуальність проблеми та розкрила потенціал акселерометра для біометричної безпеки. Використання відповідного відкритого набору даних, який містить записи з акселерометра в трьох осях, і вейвлет аналізу, дозволило зконвертувати часові ряди даних у фотозображення, які називають скалеограмами. Ці зображення потім були використані для навчання глибинних нейронних мереж.

У розділі 2 досліджено математичні основи, на яких базується робота. Було досліджено стандартні штучні нейронні мережі (ANN), рекурентні нейронні мережі (RNN), зокрема довготривалу пам'ять (LSTM), та згорткові нейронні мережі (CNN) з метою ефективної обробки послідовних даних та розпізнавання залежностей в них. Дослідження різних оптимізаторів та метрик якості дозволило знайти оптимальні параметри моделі та оцінити їх ефективність.

Додатково було розглянуто різні методи перетворення сигнальних даних у зображення – перетворення Фур’є, Габора і вейвлет-перетворення.

У третьому розділі описано, як було побудовано і протестовано багато нейронних мереж різних архітектур, зокрема ResNet, DenseNet та запропоновано нейронну мережу на основі EfficientNet_B0, яка справилася із завданням дослідження краще за всіх інших «кандидатів».

У четвертому розділі проведено функціонально-вартісний аналіз розробленого програмного. Аналіз включав оцінку вартості розробки, прогнозування користувачів і їх потреб, а також економічні вигоди від використання програмного продукту. Результати аналізу підтвердили ефективність та доцільність розробленого програмного продукту, його перспективність, конкурентоспроможність на ринку та потенційну прибутковість.

На основі проведених досліджень можна зробити наступні висновки.

1. Запропонована модель працює значно швидше й енергоефективніше за RNN та LSTM, адже має менше ваг і потребує менше обчислень під час навчання і передбачення.
2. Якість, точність ідентифікації користувача всього за 45 секунд акселерометричних даних є дуже високою, що доводить перспективність розробки комерційного рішення стосовно пасивної додаткової аутентифікації користувача смартфону за допомогою висвітленого методу. Як на мене, такий застосунок є чудовою темою для моєї магістерської роботи або навіть власного проєкту.
3. Успішне використання вейвлет-перетворення для конвертації сигнальних даних у скалеограми є науковою новизною і доводить свою перспективність. Це відкриває для нас нові варіанти вирішення подібних задач таким способом.

Отже, біометрична автентифікація користувача смартфона з використанням даних акселерометра є перспективним напрямком, оскільки акселерометр є широко розповсюдженим сенсором в сучасних смартфонах. Використавши вейвлет-перетворення для генерації зображень з «сирих» акселерометричних даних і навчивши нейронні мережі ідентифікувати користуваів за допомогою цих зображень, я підтвердив можливість використання акселерометра для аутентифікації користувача смартфона на основі його унікальних рухових характеристик. Запропонована модель показала результат у 92.53% точності на датасеті в 24600 зображень, 50 класах.

В подальшому рекомендується продовжити дослідження в напрямку використання інших біометричних параметрів разом з даними акселерометра для ще більш точної та надійної автентифікації користувачів смартфонів. Важливо також розробити методологію підвищення стійкості моделей до різних факторів, таких як різні пози користувача, шуми в даних тощо.

СПИСОК ДЖЕРЕЛ

1. «A Complete Guide to Biometric Authentication Methods» (13 квітня, 2022). Доступно за посиланням: https://ondato.com/blog/benefits-of-biometric-authentication/
2. «A Quarter of Users Don’t Understand the Risks of Mobile Cyberthreats, Kaspersky Lab Survey Shows» (26 лютого, 2015). Доступно за посиланням: https://www.kaspersky.com/about/press-releases/2015_a-quarter-of-users-don-t-understand-the-risks-of-mobile-cyberthreats-kaspersky-lab-survey-shows
3. «Non-standard smartphone wiretapping» (9 лютого, 2023). Доступно за посиланням: https://www.kaspersky.com/blog/non-standard-smartphone-wiretapping/47113/
4. «What are convolutional neural networks?» . Доступно за посиланням: https://www.ibm.com/topics/convolutional-neural-networks
5. «Accelerometer Biometric Competition. Recognize users of mobile devices from accelerometer data» (23 липня, 2013). Доступно за посиланням: https://www.kaggle.com/competitions/accelerometer-biometric-competition/overview/description
6. «Costs and benefits of authentication» (8 червня, 2021).. Доступно за посиланням: https://www.miteksystems.com/blog/costs-and-benefits-of-authentication
7. «Continuous Authentication». Доступно за посиланням: https://www.onespan.com/topics/continuous-authentication
8. «Screen time stats 2019: Here’s how much you use your phone during the workday», Jory MacKay (21 березня, 2019). Доступно за посиланням: https://blog.rescuetime.com/screen-time-stats-2018/
9. «Knowledge-based authentication (KBA) [explanation and examples]». Доступно за посиланням: https://www.incognia.com/the-authentication-reference/knowledge-based-authentication-kba-meaning-and-examples

10. «Про інноваційну технологію Face ID» (4 травня, 2022). Доступно за посиланням: https://support.apple.com/uk-ua/HT208108
11. «Human activity recognition (HAR) using machine learning». Доступно за посиланням: https://www.neuraldesigner.com/solutions/activity-recognition
12. «CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning», Aravindpai Pai (17 лютого, 2020). Доступно за посиланням: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
13. «Activation Functions in Neural Networks [12 Types & Use Cases]», Pragati Baheti (27 травня, 2021). Доступно за посиланням: https://www.v7labs.com/blog/neural-networks-activation-functions
14. «A Comprehensive Guide on Optimizers in Deep Learning», Ayush Gupta (18 травня, 2023). Доступно за посиланням: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
15. «Introduction to Wavelet Transforms», Nirdosh Bhatnagar (13 травня, 2020). Доступно за посиланням: https://www.amazon.ca/Introduction-Wavelet-Transforms-Nirdosh-Bhatnagar/dp/0367438798
16. «Everything you need to know about VGG16», Rohini G (23 вересня, 2021). Доступно за посиланням: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918
17. «Human identification based on accelerometer sensors obtained by mobile phone data», Oğuz A., Ertuğrul Ö.F. (2022). Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S174680942200369X
18. «Smartwatch-based biometric gait recognition», Johnston A.H., Weiss G.M. (2015). Доступно за посиланням: https://ieeexplore.ieee.org/document/7358794
19. «EEG-based user identification system using 1D-convolutional long short-term memory neural networks», Sun Y., Lo F.P.-W., Lo B. (2019). Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S095741741930096X

20. «Gait-based authentication using a wrist-worn device», Cola G., Avvenuti M., Musso F., Vecchio A. (2016). Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S095070512201139X#b9
21. «A novel authentication scheme based on acceleration data in WBAN», Liu B., Luo H., Chen C.W. (2017). Доступно за посиланням: https://ieeexplore.ieee.org/document/8010625
22. «Gait and respiration-based user identification using wi-fi signal», Wang X., Li F., Xie Y., Yang S., Wang Y. (2021). Доступно за посиланням: https://ieeexplore.ieee.org/abstract/document/9488277
23. «Classification of gait type based on deep learning using various sensors with smart insole», Lee S.-S., Choi S.T., Choi S.-I. (2019). Доступно за посиланням: https://www.mdpi.com/1424-8220/19/8/1757
24. «Smartphone and smartwatch-based biometrics using activities of daily living», Weiss G.M., Yoneda K., Hayajneh T. (2019). Доступно за посиланням: https://ieeexplore.ieee.org/document/8835065
25. «Towards continuous authentication on mobile phones using deep learning models», Volaka H.C., Alptekin G., Basar O.E., Isbilen M., Incel O.D. (2019). Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S187705091930941X
26. «Convolutional neural networks for user identification based on motion sensors represented as images», Benegui C., Ionescu R.T. (2020). Доступно за посиланням: https://ieeexplore.ieee.org/document/9050748
27. «Gait identification and authentication using LSTM based on 3-axis accelerations of smartphone», Watanabe Y., Kimura M. (2020). Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S1877050920318901
28. «Identification of walker identity using smartphone sensors: an experiment using ensemble learning», Angrisano A., Bernardi M.L., Cimitile M., Gaglione S., Vultaggio M. (2020). Доступно за посиланням: https://ieeexplore.ieee.org/document/8984333

29. «Learning human identity from motion patterns», Neverova N., Wolf C., Lacey G., Fridman L., Chandra D., Barbello B., Taylor G. (2016). Доступно за посиланням: https://ieeexplore.ieee.org/abstract/document/7458136
30. «Wearable device user authentication using physiological and behavioral metrics», Vhaduri S., Poellabauer C. (2017). Доступно за посиланням: https://ieeexplore.ieee.org/document/8292272
31. «Identifying users with wearable sensors based on activity patterns», ul Haq M.E., Malik M.N., Azam M.A., Naeem U., Khalid A., Ghazanfar M.A. (2020). Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S095070512201139X#b9
32. «Hold and sign: A novel behavioral biometrics for smartphone user authentication», Buriro A., Crispo B., Delfrari F., Wrona K. (2016). Доступно за посиланням: https://ieeexplore.ieee.org/abstract/document/7527780
33. «Senguard: Passive user identification on smartphones using multiple sensors», Shi W., Yang J., Jiang Y., Yang F., Xiong Y. (2011). Доступно за посиланням: https://ieeexplore.ieee.org/document/6085412
34. «Smartphone user identity verification using gait characteristics», Damaševičius R., Maskeliūnas R., Venčkauskas A., Woźniak M. (2016). Доступно за посиланням: https://www.mdpi.com/2073-8994/8/10/100
35. «Authenticating users through their arm movement patterns», Kumar R., Phoha V.V., Raina R. (2016). Доступно за посиланням: https://arxiv.org/abs/1603.02211
36. «Cell phone-based biometric identification», Kwapisz J.R., Weiss G.M., Moore S.A. Доступно за посиланням: https://ieeexplore.ieee.org/abstract/document/5634532
37. «Automated person recognition by walking and running via model-based approaches», Yam C., Nixon M.S., Carter J.N. Доступно за посиланням: https://www.sciencedirect.com/science/article/pii/S0031320303003996

38. «Gait identification using accelerometer on mobile phone», Thang H.M., Viet V.Q., Thuc N.D., Choi D. Доступно за посиланням: https://ieeexplore.ieee.org/document/6466615
39. Повний код до цього дослідження, Р. Є. Кагарлицький (червень 2023). Доступно за посиланням: https://github.com/rktraz/biometric_auth_using_wavelets

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

(Повний код див. за посиланням [39])

Файл create_scaleograms.py – конвертація «сирих» акселерометричних даних у скалеограми:

```
import pandas as pd
import pywt
import numpy as np
import matplotlib.pyplot as plt
import os
import time


def _create_scaleogram(signal: np.ndarray, n=16, wavelet="mexh") ->
np.ndarray:
    """Creates scaleogram for signal, and returns it.

    The resulting scaleogram represents scale in the first dimension, time
in
    the second dimension, and the color shows amplitude.
    """

    scale_list = np.arange(start=0, stop=len(signal)) / n + 1
    scaleogram = pywt.cwt(signal, scale_list, wavelet)[0]
    return scaleogram


started = time.time()

# for df, df_name in zip([df_train, df_test], ["train_dataset",
"test_dataset"]):
for df, df_name in zip([df_train], ["test_dataset"]):
    print(time.strftime("%H:%M:%S", time.localtime()))
    print(f"Started processing {df_name}.")
    print(32 * "*")
    print()
    # for wavelet, n in zip(["mexh", "morl"], [16, 8]):
    wavelet = "mexh"
    n = 16
    n_users = df.user_id.nunique()
    SAVE_TO = f"{df_name}_{wavelet}_method"

    # Step 1: Create windows of size 225 with a shift factor of 30
    window_size = 225
    shift_factor = 30
```

```
    # Step 2: Perform operations separately for each user
    users = df['user_id'].unique()

    for user_id in users:
        user_data = df[df['user_id'] == user_id]

        # Check if folder already exists for user_id
        output_dir = f"{SAVE_TO}/user_{user_id}"
        if os.path.exists(output_dir):
            continue
        else:
            os.makedirs(output_dir, exist_ok=True)

        # Step 3: Create scaleograms for each axis for every window
        x_axis_scaleograms = []
        y_axis_scaleograms = []
        z_axis_scaleograms = []

        for i in range(0, len(user_data) - window_size + 1, shift_factor):
            window = user_data.iloc[i:i+window_size]

            x_axis_signal = window['x_axis'].to_numpy()
            y_axis_signal = window['y_axis'].to_numpy()
            z_axis_signal = window['z_axis'].to_numpy()

            x_axis_scaleogram = _create_scaleogram(x_axis_signal, n=n,
wavelet=wavelet)
            y_axis_scaleogram = _create_scaleogram(y_axis_signal, n=n,
wavelet=wavelet)
            z_axis_scaleogram = _create_scaleogram(z_axis_signal, n=n,
wavelet=wavelet)

            x_axis_scaleograms.append(x_axis_scaleogram)
            y_axis_scaleograms.append(y_axis_scaleogram)
            z_axis_scaleograms.append(z_axis_scaleogram)

        # Step 4: Concatenate scaleograms vertically and save the images
        for i in range(len(x_axis_scaleograms)):
            scaleogram_combined = np.concatenate(
                (x_axis_scaleograms[i], y_axis_scaleograms[i],
z_axis_scaleograms[i]),
                 axis=0
            )
            plt.imshow(scaleogram_combined, aspect='auto')
            plt.axis('off')
            plt.savefig(f"{output_dir}/scaleogram_{i}.png")
            plt.close()
        print()
        print("Processed user with id", user_id)
    print(f"Processing of {SAVE_TO} folder is done. It took {time.time() -
started} sec.\n")
    print(32 * "*")
```

```
    print()
```

Файл modeling_kaggle.py – навчання різних моделей і перевірки їх ефективності (на платформі Kaggle):

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


import torch
import torchvision
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from torchvision.datasets import ImageFolder
import matplotlib.pyplot as plt

# Set device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')


def get_dataloaders(path, batch_size=32, train_size=0.8):
    # Define transformation to be applied to the scaleogram images
    transform = transforms.Compose([
        transforms.Resize((224, 224)),  # Reshape images to 224x224
        transforms.ToTensor(),           # Convert images to tensors
        transforms.Normalize((0.5,), (0.5,))  # Normalize image tensors
    ])
```

```
    # Create a dataset object
    dataset = ImageFolder(path, transform=transform)
    # Split dataset into training and validation sets
    train_size_ = int(train_size * len(dataset))
    val_size_ = len(dataset) - train_size_
    train_dataset, val_dataset = torch.utils.data.random_split(dataset,
[train_size_, val_size_])

    # Create data loaders for training and validation sets
    train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True, num_workers=2)
    val_loader = DataLoader(val_dataset, batch_size=batch_size,
shuffle=False, num_workers=2)
    return [train_loader, val_loader]


import os
import pickle
import torch
import matplotlib.pyplot as plt
import zipfile


def save_model_and_results(method: str, model, results, num_classes):
    folder_name = f'{method}_{model._get_name()}_{num_classes}classes'

    # Create the folder if it doesn't exist
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)

    train_losses, train_accuracies, val_losses, val_accuracies = results

    with open(os.path.join(folder_name, 'train_losses.pickle'), 'wb') as
file:
        pickle.dump(train_losses, file)
    with open(os.path.join(folder_name, 'val_losses.pickle'), 'wb') as file:
        pickle.dump(val_losses, file)
```

```
    with open(os.path.join(folder_name, 'train_accuracies.pickle'), 'wb') as
file:
        pickle.dump(train_accuracies, file)
    with open(os.path.join(folder_name, 'val_accuracies.pickle'), 'wb') as
file:
        pickle.dump(val_accuracies, file)


    # Save the trained model
    torch.save(model.state_dict(), os.path.join(folder_name,
f'{model._get_name()}.pth'))


    # Visualize and save plots
    epochs = range(1, len(val_losses) + 1)


    # Plot training and validation losses
    plt.figure(figsize=(10, 5))
    plt.plot(epochs, train_losses, label='Train')
    plt.plot(epochs, val_losses, label='Validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.savefig(os.path.join(folder_name, 'loss_plot.png'))
    plt.show()
    plt.close()


    # Plot training and validation accuracies
    plt.figure(figsize=(10, 5))
    plt.plot(epochs, train_accuracies, label='Train')
    plt.plot(epochs, val_accuracies, label='Validation')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.legend()
```

```
    plt.savefig(os.path.join(folder_name, 'accuracy_plot.png'))
    plt.show()
    plt.close()

    # Create a zip file of the folder
    zip_filename = f'{folder_name}.zip'
    with zipfile.ZipFile(zip_filename, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, dirs, files in os.walk(folder_name):
            for file in files:
                file_path = os.path.join(root, file)
                zipf.write(file_path, arcname=os.path.relpath(file_path,
folder_name))

    print(f"Successfully saved the model, results, and plots in
{folder_name}!")
    print(f"Created a zip file: {zip_filename}")


def train(model, num_epochs, dataloaders, criterion, optimizer):
    train_loader, val_loader = dataloaders
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # Initialize lists to store training and validation metrics
    train_losses = []
    train_accuracies = []
    val_losses = []
    val_accuracies = []

    for epoch in range(num_epochs):
        # Training
        model.train()
        train_loss = 0.0
        train_correct = 0

        for images, labels in train_loader:
            images = images.to(device)
```

```
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = model(images)
        loss = criterion(outputs, labels)

        loss.backward()
        optimizer.step()

        train_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        train_correct += (predicted == labels).sum().item()

    train_loss /= len(train_loader.dataset)
    train_accuracy = 100.0 * train_correct / len(train_loader.dataset)

    # Validation
    model.eval()
    val_loss = 0.0
    val_correct = 0

    with torch.no_grad():
        for images, labels in val_loader:
            images = images.to(device)
            labels = labels.to(device)

            outputs = model(images)
            loss = criterion(outputs, labels)

            val_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            val_correct += (predicted == labels).sum().item()
```

```
        val_loss /= len(val_loader.dataset)
        val_accuracy = 100.0 * val_correct / len(val_loader.dataset)

        # Update metrics
        train_losses.append(train_loss)
        train_accuracies.append(train_accuracy)
        val_losses.append(val_loss)
        val_accuracies.append(val_accuracy)

        print(f'Epoch {epoch + 1}/{num_epochs}: '
              f'Train Loss: {train_loss:.4f}, Train Accuracy:
{train_accuracy:.2f}%, '
              f'Val Loss: {val_loss:.4f}, Val Accuracy:
{val_accuracy:.2f}%')
        print(time.strftime("%H:%M:%S", time.localtime()))
    return [train_losses, train_accuracies, val_losses, val_accuracies]


train_dataset_path = "/kaggle/input/new-wavelet-datasets-for-my-
diploma/train_dataset_mexh_method/train_dataset_mexh_method"
test_dataset_path = "/kaggle/input/new-wavelet-datasets-for-my-
diploma/test_dataset_mexh_method 2/test_dataset_mexh_method"


import time

NUM_EPOCHS = 25

for dataset_path in [train_dataset_path]: # in case you want to train
different models on different datasets
    print("\n\n" + 64*"*")
    print(time.strftime("%H:%M:%S", time.localtime()))
    print("Processing", dataset_path.split("/")[-1], "data")
    print("\n\n" + 64*"*" + "\n\n")
    dataloaders = get_dataloaders(dataset_path, batch_size=32,
train_size=0.8)

    # Load the pre-trained efficientnet_b0 model
    model = torchvision.models.efficientnet_b0(pretrained=True)
```

```
    # Modify the classifier for the number of classes in your dataset
    num_classes = len(dataloaders[1].dataset.dataset.classes)
    model.classifier[1] = nn.Linear(1280, num_classes)

    # Move the model to the device
    model = model.to(device)

    # Define the loss function
    criterion = nn.CrossEntropyLoss()

    # Define the optimizer
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    results = train(model=model, num_epochs=NUM_EPOCHS,
dataloaders=dataloaders,
                    criterion=criterion, optimizer=optimizer)
    method = dataset_path.split("/")[-1].split("_")[-2]
    save_model_and_results(method, model, results, num_classes)

    print(time.ctime())
    print("This iteration ended successfully.")


transform = transforms.Compose([
    transforms.Resize((224, 224)),  # Reshape images to 224x224
    transforms.ToTensor(),           # Convert images to tensors
    transforms.Normalize((0.5,), (0.5,))  # Normalize image tensors
])

# Create a dataset object
test_dataset = ImageFolder(test_dataset_path, transform=transform)

test_dataloader = DataLoader(test_dataset, batch_size=32, shuffle=False,
num_workers=2)
```

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix


# Set the model to evaluation mode

model.eval()


# Load the saved model state dictionary

#
model.load_state_dict(torch.load("/kaggle/working/mexh_EfficientNet_50classes/EfficientNet.pth"))

model.load_state_dict(torch.load("/kaggle/input/new-wavelet-datasets-for-my-diploma/mexh_EfficientNet_50classes/EfficientNet.pth"))


# Test the model on the new data

total_correct = 0

total_samples = 0

all_predictions = []

all_labels = []


with torch.no_grad():

    for inputs, labels in test_dataloader:

        inputs = inputs.to(device)  # Move inputs to the appropriate device (e.g., GPU)

        labels = labels.to(device)  # Move labels to the appropriate device


        outputs = model(inputs)  # Forward pass

        _, predicted = torch.max(outputs, dim=1)  # Get the predicted labels


        total_samples += labels.size(0)  # Increment the total number of samples

        total_correct += (predicted == labels).sum().item()  # Increment the correct predictions


        all_predictions.extend(predicted.cpu().numpy())  # Collect all predicted labels
```

```
        all_labels.extend(labels.cpu().numpy())  # Collect all ground truth
labels

accuracy = total_correct / total_samples
print(f"Test Accuracy: {accuracy:.2%}")

# Create confusion matrix
confusion = confusion_matrix(all_labels, all_predictions)

# Display confusion matrix
plt.figure(figsize=(8, 8))
plt.imshow(confusion, cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.colorbar()

# Add labels to each cell in the confusion matrix
num_classes = len(class_names)
tick_marks = np.arange(num_classes)
plt.xticks(tick_marks, class_names, rotation=90)
plt.yticks(tick_marks, class_names)

# Add count values to the cells
thresh = confusion.max() / 2.
for i in range(num_classes):
    for j in range(num_classes):
        plt.text(j, i, format(confusion[i, j], 'd'),
horizontalalignment="center",
                 color="white" if confusion[i, j] > thresh else "black")

plt.tight_layout()
plt.show()
```