ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»

Факультет комп'ютерно-інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету)

Кафедра прикладної математики та інформатики
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри ПМІ
Наталія Маслова
(підпис) (ініціали, прізвище)
«____»_____2023 р.

Випускна кваліфікаційна робота

магістра
(освітньо-кваліфікаційний рівень)

на тему: «Нейромережева аутентифікація користувачів за клавіатурним почерком»

Виконала: студентка 2 курсу, групи ІПЗм-22
(шифр групи)

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва напряму підготовки)

Єжова Є. О. _____
(прізвище та ініціали) (підпис)

Керівник проф. каф. ПМІ, д.т.н., Башков Є. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент зав. каф. АТ, к.т.н., доц. Поцепаєв В.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент в.о. зав. каф. ЕТ, к.т.н., доц. Ковальов С. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій випускній кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Луцьк – 2023

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь магістр
Спеціальність (напрям підготовки) 121 Інженерія програмного забезпечення

(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМІ
_____ /Н. О. Маслова/
«____»_____2023 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Єжовій Єлизаветі Олексіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Нейромережева аутентифікація користувачів за клавіатурним почерком
керівник роботи проф. каф. ПМІ, д.т.н., Башков Євген Олександрович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом від « 22 » вересня 20 23 року № 478
2. Строк подання студентом роботи 5 грудня 2023 року
3. Вихідні дані до роботи результати переддипломної практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) огляд методів аутентифікації за клавіатурним почерком; 2) аналіз нейронних мереж для аутентифікації за клавіатурним почерком; 3) вибір середи розробки та типу архітектури нейронної мережі; 4) проєктування архітектури нейронної мережі; 5) опис процесу збору даних клавіатурного почерку; 6) підбір параметрів та навчання нейронної мережі; 7) тестування, аналіз результатів та визначення показників точності мережі
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) робота містить 59 рисунків, в цілому в кількості, достатній для демонстрації матеріалів випускної кваліфікаційної роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Назарова І.А.		

7. Дата видачі завдання 4 жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналітичний огляд предметної області, дослідження теоретичної бази, вивчення джерел інформації	02.10.23-15.10.23	
2	Аналіз сучасних методів аутентифікації за клавіатурним почерком, вибір інструментів розробки	16.10.23-31.10.23	
3	Проєктування і розробка архітектури нейронної мережі для аутентифікації користувача за клавіатурним почерком	01.11.23-19.11.23	
4	Тестування та оцінка розробленої мережі	20.11.23-30.11.23	
5	Оформлення пояснювальної записки та опис створеної мережі. Проходження нормоконтролю	01.12.23-05.12.23	
6	Перевірка пояснювальної записки антиплагіатною системою. Оформлення презентації до роботи, графічного матеріалу та рецензування роботи	06.12.23-19.12.23	
7	Захист роботи	20.12.2023	

Студент _____ Єжова Є. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Башков Є. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Єжова Є. О. Нейромережева аутентифікація користувачів за клавіатурним почерком / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 121 Інженерія програмного забезпечення. – ДВНЗ ДонНТУ, Луцьк, 2023.

Випускну магістерську роботу присвячено дослідженню наявних нейромережевих методів аутентифікації користувачів на основі аналізу їх клавіатурного почерку для створення мережі, яка дозволяє впізнавати користувачів за їхнім унікальним стилем введення тексту на клавіатурі.

Об’єкт дослідження даної роботи – процес аутентифікації користувачів в комп’ютерно-інформаційних системах.

Предмет дослідження даної роботи – нейромережевий метод аутентифікації користувачів за клавіатурним почерком.

Мета роботи – підвищення рівня захисту комп’ютерно-інформаційних систем шляхом високоточної автоматичної ідентифікації користувачів за їх клавіатурним почерком за допомогою нейронної мережі аутентифікації, що забезпечить ідентифікацію користувача на основі унікальних характеристик його введення на клавіатурі.

Задачами даної роботи є проведення порівняльного аналізу наявних методів аутентифікації користувачів за клавіатурним почерком, зокрема, з використання нейронних мереж, для подальшого обрання архітектури, яка забезпечує високу точність в завдання аутентифікації за клавіатурним почерком, проведення налаштування, навчання та тестування нейронної мережі, використовуючи відповідні методи оптимізації, щоб досягнути максимальної точності моделі та оцінка точності розробленої архітектури мережі на наборі тестових даних, для оцінки її здатності розрізняти аутентичний почерк від підробленого.

Результатом роботи стала нейронна мережа для аутентифікації користувачів за клавіатурним почерком.

Наукова новизна полягає в отриманні більш точної архітектури нейронної мережі, яка поєднує згорткові нейронні мережі та довгу короткочасну пам’ять, що дозволяє підвищити рівень точності процесу аутентифікації. Практичне значення полягає в накопиченні рекомендацій щодо розробки та навчання згорткових нейронних мереж та довгої короткочасної пам’яті в цілях використання в аутентифікації користувачів на основі їх клавіатурного почерку.

Ключові слова: аутентифікація, нейронна мережа, точність, клавіатурний почерк, архітектура, Python, TensorFlow, KerasTuner, LSTM, CNN.

Список публікацій здобувача:

1. Башков Є. О., Алтухова Т.В., Єжова Є. О. Розробка методу аутентифікації користувача на основі клавіатурного почерку / Є.О. Башков, Т.В. Алтухова, Є.О. Єжова // Наукові праці Донецького національного технічного університету. – Луцьк, 2023. - №2(35)’2022 - №1(36)’2023. – С. 61-68. DOI: 10.31474/1996-1588-2023-1-36-61-68

2. Єжова Є. О. Використання нейронних мереж для аутентифікації користувача за клавіатурним почерком / Є. О. Єжова // Моделювання і комп’ютерна графіка: зб. матер. Восьмої міжнар. наук.-техн. конф, 11-14 квітня 2023 р.: присв. 100-річчю з дня народж. Л. П. Фельдмана / Донецький національний технічний університет. – Луцьк – Київ: ДВНЗ «ДонНТУ», 2023. – С. 66-72

3. Єжова Є. О. Аналіз застосування сучасних методів розпізнавання облич з використанням нейронних мереж / Є. О. Єжова // «ТАК»: телекомунікації, автоматика, комп’ютерно-інтегровані технології: зб.

доповідей Всеукр. наук.-практ. конф. молодих вчених, 1 грудня 2022 р. / ДВНЗ «ДонНТУ; відп. ред. Г.В. Ступак. – Луцьк: ДВНЗ «ДонНТУ», 2022. – С. 7-10

4. Єжова Є. О. Overview and problems of biometric authentication methods / Є. О. Єжова // Актуальні питання та перспективи проведення наукових досліджень: матеріали III Міжнародної студентської наукової конференції (Т. 3), м. Трускавець, 10 грудня, 2021 рік / ГО «Молодіжна наукова ліга». – Вінниця: ГО «Європейська наукова платформа», 2021. – С. 53–55 с. DOI 10.36074/liga-inter-10.12.2021

Список публікацій прийнятих до друку:

1. Єжова Є. О. Нейронна мережа аутентифікації користувача за клавіатурним почерком / Є. О. Єжова // Електронні інформаційні ресурси: створення, використання, доступ: Зб. Матеріалів Міжнародної нак.-практ. Інтернет конф. (2023).

ABSTRACT

Yezhova Y. Users' neural network authentication by keyboard dynamic / Graduation qualification work for obtaining an educational degree “Master” in specialty 121 “Software Engineering”. – DonNTU, Lutsk, 2023.

This master's thesis is devoted to the study of existing neural network methods of user authentication based on the analysis of their keyboard dynamics to create an effective network that allows users to recognize their unique style of typing.

The object of research of this paper is the process of user authentication in computer information systems.

The subject in this work is a neural network method of user authentication by keyboard handwriting.

The aim of the work is to increase the level of protection of computer information systems by means of high-precision automatic identification of users by their keyboard dynamic using a neural authentication network, which will ensure the identification of the user based on the unique characteristics of his keyboard input.

The tasks of this work are to conduct a comparative analysis of existing methods of user authentication by handwriting, in particular, using neural networks, to further select an architecture that provides high accuracy in the task of authentication by handwriting, to set up, train and test a neural network using appropriate optimization methods to achieve maximum model accuracy, and to evaluate the accuracy of the developed network architecture on a set of test data to assess its ability to distinguish between.

The result of the work is a neural network for user authentication by keyboard dynamic.

The scientific novelty consists in obtaining a more efficient neural network architecture that combines convolutional neural networks and long short-term memory, increases the level of accuracy of the authentication process.

The practical significance is to accumulate recommendations for the development and training of convolutional neural networks and long short-term memory for use in user authentication based on their keyboard dynamics.

Keywords: authentication, neural network, accuracy, keyboard dynamic, architecture, Python, TensorFlow, KerasTuner, LSTM, CNN.

Publisher’s publication list:

1. Bashkov Y.O., Altukhova T.V., Yezhova Y.O. Development of a user authentication method based on keyboard dynamic / Y.O. Bashkov, T.V. Altukhova, Y.O. Yezhova // Scientific works of Donetsk National Technical University. - №2(35)'2022 - №1(36)'2023. - C. 61-68. DOI: 10.31474/1996-1588-2023-1-36-61-68

2. Yezhova Y. O. Using neural networks for user authentication by keyboard dynamic / Y. O. Yezhova // Modeling and computer graphics: a collection of materials. Eighth International Scientific and Technical Conference, April 11-14, 2023: dedicated to the 100th anniversary of L.P. Feldman / Donetsk National Technical University - Lutsk - Kyiv: DonNTU, 2023 - P. 66-72.

3. Yezhova Y.O. Analysis of the application of modern methods of face recognition using neural networks / Y.O. Yezhova // TAC": telecommunications, automation, computer-integrated technologies: a collection of reports of the All-Ukrainian scientific and practical conference of young scientists, December 1, 2022 / SHEI "DonNTU; editor-in-chief G.V. Stupak: SHEI "DonNTU", 2022. - P. 7-10.

4. Yezhova Y.O. Overview and problems of biometric authentication methods / Y.O. Yezhova // Topical issues and prospects of scientific research: materials of the III International Student Scientific Conference (Vol. 3), Truskavets, December 10, 2021 / NGO "Youth Scientific League." - Vinnytsia: NGO "European Scientific Platform", 2021. - P. 53-55. DOI 10.36074/liga-inter-10.12.2021

Publisher`s list of publications accepted for publication:

1. Yezhova Y. Neural network of user authentication by keyboard handwriting / Y. Yezhova // Electronic information resources: creation, use, access: Proceedings of the International Scientific and Practical. Internet Conf. (2023).

ЗМІСТ

Вступ 13
Розділ 1 Огляд методів аутентифікації за клавіатурним почерком 15
1.1.1 Гауссова модель суміші 15
1.1.2 Дискримінантний аналіз 17
1.2 Динамічні підходи 17
1.2.1 Прихована модель Маркова 17
1.2.2 Динамічне часове вікно 18
1.3 Нейронні мережі для аутентифікації користувача за клавіатурним почерком 20
1.3.1 Загальний огляд нейронних мереж 20
1.3.2 Архітектури нейронних мереж, що використовуються в завданні аутентифікації за клавіатурним почерком 37
1.4 Визначення мети та основних задач роботи 43
1.5 Висновок за розділом 44
Розділ 2 Архітектура нейронної мережі аутентифікації користувача за клавіатурним почерком 45
2.1 Вибір середи розробки 46
2.2 Структура моделі 47
2.3 Висновок за розділом 52
Розділ 3 Підготовка даних для навчання мережі аутентифікації користувача за клавіатурним почерком 54
3.1 Опис процесу збору даних клавіатурного введення 54
3.2 Етапи попередньої обробки для підготовки даних 59
3.3 Висновок за розділом 68
Розділ 4 Результати експериментів з навчання та тестування 70
4.1 Процес навчання та налаштування параметрів 70
4.2 Тестування та оцінювання 75

4.3 Виклад результатів 80
4.4 Висновок за розділом 81
Висновок 82
Список використаних джерел 83
Додаток А Зауваження нормоконтролера 89
Додаток Б Фрагметр лістингу програмного коду 90
Додаток В Матеріали презентації 107

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС – інформаційна система

AUC – area under the ROC-curve

CNN – convolutional neural networks

DA – discriminant analysis

DTW – dynamic time warping

FA – activation function

FAR – false acceptance rate

FN – false negative

FNN – feed-forward neural networks

FP – false positive

FPR – false positive rate

FRR – false rejection rate

GMM – gaussian mixture model

GRU – gated recurrent unit

HMM – hidden Markov model

KD – keystroke dynamics

LSTM – long short-term memory

NN – neural network

ReLU – rectified liner unit

RMSprop – root mean square propagation

RNN – recurrent neural networks

ROC-крива – receiver operating characteristic

Softmax – багатозмінна логістична функція

TN – true negative

TP – true positive

ВСТУП

В епоху, коли інформаційні системи (далі – ІС) вкрай важливі для нашого повсякденного життя та бізнесових процесів, безпека та аутентифікація користувачів стають однією з головних проблем. Магістерська робота на тему «Нейромережева аутентифікація користувачів за клавіатурним почерком» стає надзвичайно актуальною і важливою в цьому контексті.

Сьогодні ІС містять величезні обсяги особистих і конфіденційних даних, і їх безпека — це завдання першості. Традиційні методи аутентифікації, такі як паролі, вже не забезпечують належного рівня безпеки, оскільки легко піддаються атакам та вимагають від користувачів запам'ятовування десятків чи навіть сотень різних паролів.

Нейромережева аутентифікація на основі клавіатурного почерку (keystroke dynamics, далі – KD) використовує унікальні риси почерку для ідентифікації користувачів. Кожна особистість має свої унікальні особливості у способі, яким вона набирає текст на клавіатурі — натиск на клавіші, паузи між натисками. Це робить нейромережеву аутентифікацію надійною та складною до підробки. Важливо враховувати, що розробка та впровадження нейромережевої аутентифікації в комп'ютерно-інформаційних системах потребує додаткового дослідження та оптимізації. Проте, ця технологія має великий потенціал для підвищення рівня безпеки в ІС та зручності для користувачів. Таким чином, магістерська робота на цю тему може внести важливий внесок у розвиток інформаційних технологій, зробити ІС безпечнішими та зручнішими для користувачів, а також відповісти на сучасні виклики щодо кібербезпеки та захисту особистих даних в цифровому світі.

Метою магістерської роботи є підвищення рівня захисту комп’ютерно-інформаційних систем шляхом високоточної автоматичної ідентифікації користувачів за їх KD за допомогою нейронної мережі (neural network, далі –

NN) аутентифікації, що забезпечить ідентифікацію користувача на основі унікальних характеристик його введення на клавіатурі.

Об’єкт дослідження даної роботи є процес аутентифікації користувачів в комп’ютерно-інформаційних системах.

Предмет дослідження даної роботи є нейромережевий метод аутентифікації користувачів за KD.

Для досягнення поставленої мети, необхідно розв’язати такі задачі: проведення порівняльного аналізу наявних методів аутентифікації користувачів за KD, зокрема, з використання NNs, для подальшого обрання архітектури, яка забезпечує високу точність в завдання аутентифікації за KD, проведення налаштування, навчання та тестування NN, використовуючи відповідні методи оптимізації, щоб досягнути максимальної точності моделі та оцінка точності розробленої архітектури мережі на наборі тестових даних, для оцінки її здатності розрізняти аутентичний почерк від підробленого.

Наукова новизна полягає в отриманні більш точної архітектури NN, яка поєднує згорткові NNs (convolutional neural networks, далі – CNN) та довгу короткочасну пам’ять (long short-term memory, далі – LSTM), що дозволяє підвищити рівень точності процесу аутентифікації.

Практичне значення полягає в накопиченні рекомендацій щодо розробки та навчання CNN та LSTM в цілях використання в аутентифікації користувачів на основі їх KD.

Результатом роботи стала NN для аутентифікації користувачів за KD.

РОЗДІЛ 1
ОГЛЯД МЕТОДІВ АУТЕНТИФІКАЦІЇ ЗА КЛАВІАТУРНИМ ПОЧЕРКОМ

Аутентифікація за KD здійснюється методом біометричної ідентифікації користувача на основі аналізу його унікального способу набору тексту на клавіатурі [1].

Під час аутентифікації збираються дані про час, який затрачує користувач на набір певного тексту, а також про темп, ритм та інші параметри набору. Для порівняння збережених даних з даними користувачами використовуються різні алгоритми та моделі, включаючи NNs. Аутентифікація за KD може бути ефективним методом ідентифікації користувачів, особливо у випадках, коли інші методи аутентифікації, такі як паролі, не є достатньо надійними [2].

В науковій літературі існує велика кількість різноманітних методів, які беруться за основу аутентифікації користувача.

Далі ми роздивимось такі підходи:

- на основі статистичних методів;
- на основі динамічних методів;
- засновані на машинному навчанні.

Нижче наведено огляд кожного з вищезгаданих методів.

1.1 Статистичні методи

Ці методи засновані на зборі статистичних даних про KD користувача і порівнянні цих даних із заздалегідь визначеними шаблонами.

1.1.1 Гаусова модель суміші

Одним із найпоширеніших статистичних методів для моделювання KD є гаусова модель суміші (gaussian mixture model, далі – GMM). GMM – це параметрична функція щільності, яка складається зі зваженої суми гаусових

компонент. Кожна компонента в GMM представляється вектором середніх значень і матрицею коваріацій, яка включає дисперсії компонентів і коваріації між ними.

При застосуванні GMM для моделювання KD, зазвичай використовуються декілька компонент, які відповідають різним характеристикам почерку. Наприклад, можуть бути використані компоненти, що відповідають швидкості писання, натисканню клавіш, часовим інтервалам між натисканнями тощо [3].

GMM дає змогу представляти дані про KD у вигляді суміші гаусівських розподілів. Для аутентифікації користувача використовується метод максимальної правдоподібності, який дає змогу визначити, наскільки ймовірно, що користувач є власником облікового запису, порівнюючи його дані про KD із заздалегідь визначеними шаблонами.

Важливим аспектом використання GMM для моделювання KD є збір якісних даних для тренування. Чим більш репрезентативний тренувальний набір даних, тим точніша може бути модель GMM. Крім того, для покращення точності розпізнавання можуть бути застосовані додаткові техніки, такі як нормалізація даних, видалення шуму, використання інших методів класифікації (наприклад, NNs) або комбінування GMM з іншими моделями.

Даний метод має свої переваги та недоліки. GMM може моделювати складні розподіли клавіатурних позначок за допомогою гауссових компонент, що дозволяє точніше враховувати варіації в KD користувачів, та може розпізнавати невідомих користувачів, які не були використані під час тренування моделі, що дозволяє виявляти потенційних зловмисників.

Однак, її використання вимагає достатньо великого та репрезентативного тренувального набору даних для побудови точної моделі, а також обчислення параметрів GMM і оцінки ймовірностей вхідних даних вимагає потужних обчислювальних ресурсів, особливо при роботі з великими обсягами даних [4].

1.1.2 Дискримінантний аналіз

Ще один підхід, заснований на статистичних методах – це метод дискримінантного аналізу (discriminant analysis, далі – DA), який використовується для побудови моделі KD користувача на основі статистичних даних. DA визначає межі, що розділяють дані про KD користувачів, і використовує їх для класифікації користувачів як істинних або підробок.

DA, на відміну від GMM, є простішим та менш вимогливим до ресурсів методом, оскільки він базується на простому вимірюванні часу. Але він не такий гнучкий, як GMM, при моделюванні складних розподілів KD. Він може бути менш точним у врахуванні варіацій в KD користувачів. Також DA потребує побудови статистичних моделей для кожного користувача. Це може вимагати додаткового часу та зусиль при тренуванні моделі та вимагати багаторазового оновлення моделі при змінах в KD користувача [5].

Кінцевий вибір між GMM і DA залежить від конкретних потреб і обмежень системи, а також від якості доступних тренувальних даних і обчислювальних ресурсів.

1.2 Динамічні підходи

Динамічні підходи в аутентифікації на основі KD використовують послідовності часових рядів, отриманих від користувачів під час набору тексту. Вони враховують не тільки статистичні параметри, а й динамічні характеристики, такі як зміни швидкості набору, тиску на клавіші тощо.

Один із динамічних підходів – це підхід на основі прихованих моделей Маркова (hidden Markov model, далі – HMM).

1.2.1 Прихована модель Маркова

HMMs є одним з найпоширеніших математичних апаратів, що використовується для багатьох класифікаторів та моделювання різноманітних проблем.

В алгоритмі, що наведено в [6] використовується набір даних динаміки натискання клавіш, отриманий від групи користувачів, які вводять набір заздалегідь визначених фраз. При попередній обробці даних витягується тривалість натискання клавіш, а потім використовуються HMMs для моделювання послідовностей ознак. HMMs навчаються на підмножині даних та використовуються для класифікації решти даних як належних законному користувачеві чи самозванцю.

HMMs можуть бути корисним інструментом для аутентифікації користувача на основі динаміки натискання клавіш та використовуватися в різних програмах, таких як контроль доступу, системи безпеки та комп'ютерній експертизі.

Однією з ключових переваг використання HMMs є їх здатність моделювати послідовність подій, що постійно змінюються. В KD тривалість натискання клавіш може варіюватися, і HMMs можуть використовуватися для захоплення цих залежностей та моделювання динаміки натискання клавіш. Ще однією перевагою HMMs є їх здатність до навчання на тренувальних даних та класифікації нових даних [6].

Однак, для успішного використання HMMs потрібно мати достатньо даних, що представляють KD різних користувачів. Крім того, правильний вибір параметрів та налаштування моделей HMMs може вимагати досвіду та експертизи [6].

Інший динамічний підхід – це підхід на основі динамічного часового вікна (dynamic time warping, далі – DTW).

1.2.2 Динамічне часове вікно

Алгоритм DTW використовується для обчислення подібності між двома часовими рядами шляхом знаходження оптимального вирівнювання між ними. У динамічній аутентифікації натискання клавіш даний алгоритм використовується для порівняння часу натискання клавіш користувачем із часом попередньо записаних натискань клавіш для цього користувача.

Алгоритм враховує варіації швидкості введення, паузи між натисканнями клавіш та інші фактори, які можуть впливати на часові шаблони.

DTW передбачає динамічне налаштування часового вікна, яке використовується для обчислення відстані DTW між шаблонами часу натискання клавіші користувача та збереженим еталонним шаблоном. Часове вікно динамічно регулюється залежно від тривалості введення користувача, замість використання фіксованого часового вікна для всіх користувачів.

В [7] використовується DTW для вимірювання подібності між справжніми та тестовими значеннями й стверджується, що DTW є більш точним методом, ніж традиційні методи евклідової відстані, оскільки він враховує часове вирівнювання характеристик підпису.

Щоб оцінити цей метод, в [7] проводились експерименти, використовуючи загальнодоступний набір вхідних даних, в яких порівнювалась продуктивність розробленого методу з кількома іншими широко використовуваними методами аутентифікації, включаючи евклідову відстань, манхеттенську відстань і косинусну подібність. Результати показали, що запропонований метод перевершує інші методи з точки зору точності.

Загалом у [7] представлено підхід до аутентифікації в автономному режимі, який поєднує статистичний аналіз і DTW для досягнення високої точності.

Основною перевагою DTW є його здатність враховувати варіації у часових шаблонах, DTW здатний знаходити оптимальне вирівнювання між двома часовими рядами, враховуючи ці варіації.

Однак, використання DTW також має свої недоліки. Воно може бути обчислювально витратним, особливо для великих масивів даних або складних шаблонів. Крім того, воно може бути чутливим до шуму та інших джерел помилок, оскільки враховує всі деталі в часових рядах.

На підставі аналізу [7-11] можна зробити висновок про відсутність загальноприйнятої методики генерації вхідних даних для модулів розпізнавання KD. Однак більшість наявних підходів мають спільну

характеристику, яка полягає у використанні відфільтрованих значень часу між натисканнями окремих клавіш як вхідних даних. Крім того, існує кореляція між вибором вхідних даних і математичною структурою, яка використовується в процесі розпізнавання, яка передбачає порівняння послідовності введення з попередньо визначеним шаблоном KD. Розрізняють два типи моделей розпізнавання: на основі заздалегідь визначених фрагментів тексту та на основі фрагментів тексту з довільним вмістом. В обох випадках оператор повинен кілька разів ввести один або кілька фрагментів одного тексту для встановлення стандартів рукописного введення на клавіатурі.

Проте [12, 13] продемонстрували обмежену адаптивність статистичних моделей до варіацій емоційного стану оператора та складність їх формування при аналізі довільних текстових фрагментів, а також вказано, що застосування NNs є перспективним напрямком для вдосконалення систем розпізнавання користувача за KD.

1.3 Нейронні мережі для аутентифікації користувача за клавіатурним почерком

Одним з основних підходів динамічної аутентифікації користувача за KD є використання NNs для вивчення та розпізнавання шаблонів введення.

Щоб бути точними, ці методи зазвичай потребують великої кількості навчальних даних, але вони можуть досягти високого рівня точності та надійності.

NNs можуть обробляти послідовність вхідних даних, наприклад, натискань клавіш, і використовувати інформацію про попередні входи для прийняття рішення про те, чи є поточний користувач легітимним.

1.3.1 Загальний огляд нейронних мереж

NNs – це комп'ютерні моделі, які імітують роботу людського мозку. Вони складаються з багатьох взаємопов'язаних нейронів, які обробляють вхідні дані та генерують відповідний вихідний сигнал. NNs використовуються

в різних областях, таких як обробка природної мови, зображення, комп'ютерного зору, аналіз даних тощо [14].

Однією з головних особливостей NNs є їх здатність навчатись на основі досвіду. NNs можуть вивчити багато даних та використовувати цей досвід для підвищення їх точності. Вони можуть виявити складні нелінійні залежності між вхідними та вихідними даними та прогнозувати результати від нових даних. Навчання NNs може зайняти багато часу та вимагати великої кількості даних та обчислювальних ресурсів.

Для підвищення якості навчання та зменшення обчислювальних витрат нерідко використовуються попередньо навчені моделі. Цей підхід, відомий як попереднє навчання (pre-training), передбачає навчання моделі на великому наборі даних, наприклад, на загальнодоступних даних, а потім настройку моделі на конкретну задачу з меншою кількістю тренувальних даних.

На точність роботи навченої моделі також впливає правильний вибір функції активації (activation function, далі – FA) та функції втрат (loss function).

FA спеціально використовуються в NNs для перетворення вхідного сигналу на вихідний, який, своєю чергою, подається на вхід наступного шару в стеку. У NN обчислюється сума добутків вхідних сигналів та відповідні їм ваги й далі до неї застосовується FA, щоб отримати вихідний сигнал цього шару і подати його як вхідний сигнал наступному шару.

Точність прогнозування NN визначається типом використовуваної FA. Найчастіше використовуються нелінійні FA. NN працює так само як і лінійна регресійна модель, де прогнозований вихід дорівнює вхідним даним, якщо FA не визначена. Те ж саме відбувається, якщо використовується лінійна FA, де вихід подібний до входу, поданого разом з деякою похибкою.

Лінійна FA має лінійну границю, і якщо вона використовується, то мережа може адаптуватися лише до лінійних змін вхідних даних, але в реальному світі помилки мають нелінійні характеристики, які, своєю чергою, впливають на здатність NNs навчатися на помилкових даних. Тому нелінійні FA є кращими за лінійні FA в NN [15].

Якщо в NN не використовується FA, то вихідний сигнал буде просто простою лінійною функцією, тобто поліномом першого степеня. Хоча лінійні рівняння прості та легко розв'язуються, їх складність обмежена, і вони не мають здатності навчатися і розпізнавати складні відображення з даних. NN без FA в більшості випадків діє як лінійна регресійна модель з обмеженою продуктивністю і потужністю.

Функції, які мають ступінь більше одиниці та мають кривизну при побудові графіків, називаються нелінійними функціями. Від NN вимагається вивчати, представляти й обробляти будь-які дані й будь-яку довільну складну функцію, яка відображає перехід з входи у виходи. NNs також відомі як універсальні апроксиматори функцій, що означає, що вони можуть обчислювати та вивчати будь-яку функцію, яка їм надається [15].

Таким чином, необхідно застосувати FA, щоб зробити мережу динамічною і додати їй здатність витягувати складну і заплутану інформацію з даних і представляти нелінійні заплутані випадкові функціональні залежності між входом і виходом. Отже, додаючи мережі нелінійності за допомогою нелінійних FA, можна досягти нелінійних відповідностей між входами та виходами.

Важливою особливістю активаційної функції є те, що вона повинна бути диференційованою для реалізації стратегії оптимізації зворотного поширення, щоб обчислити помилки або втрати щодо ваг і, зрештою, оптимізувати ваги, використовуючи техніку оптимізації для зменшення помилок [16].

Далі описано деякі з таких відомих FA. Сигмоїдальна функція (sigmoid activation function) – одна з найбільш широко використовуваний нелінійних FA. Оскільки бінарні моделі прогнозування мають класифікаційний вихід або 0, або 1, сигмоїдальні функції перетворюють вхідний сигнал у діапазон значень між 0 і 1, який потім може бути пороговим залежно від вимог.

Її можна визначити за формулою 1.1.

$$f(x) = \frac{1}{1 + e^{-x}}. \quad (1.1)$$

Сигмоїдна функція не є симетричною щодо нуля, що означає, що знаки всіх вихідних значень нейронів будуть однаковими. Ця проблема може бути вирішена шляхом масштабування сигмоїдальної функції.

Крім того, сигмоїдальна функція є безперервно диференційованою і гладкою S-подібною функцією, що безперервно диференціюється [17].

Графік даної функції зображено на рисунку 1.1.

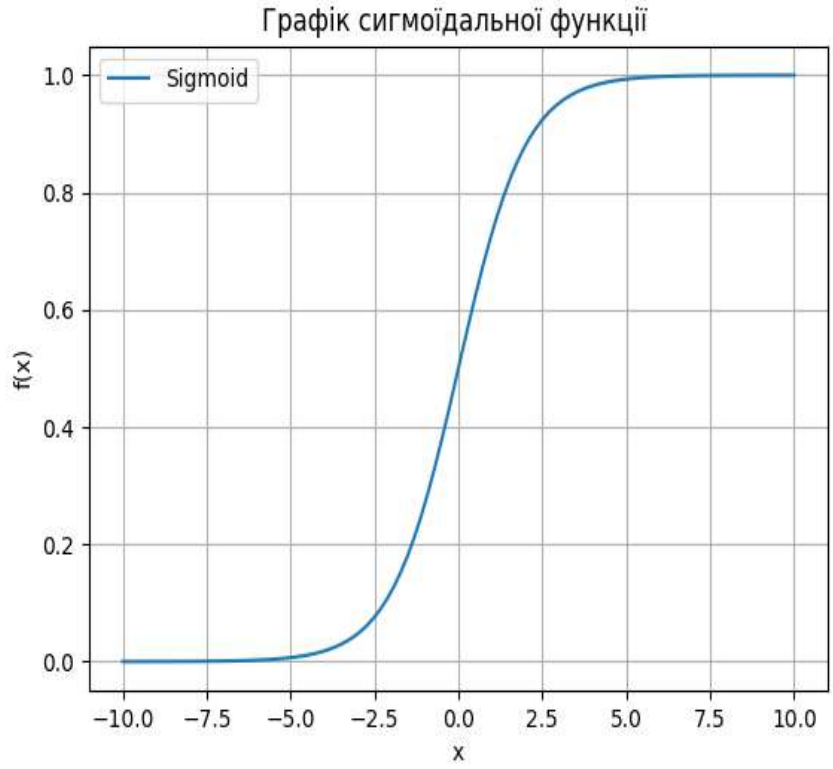


Рисунок 1.1 – Графік сигмоїдальної функції

Джерело: за даними [17]

Недоліком є те, що значення f(x) менше реагує на зміну x при наближенні до обох кінців сигмоїди. Це призводить до малих значень градієнта в цих областях, що, своєю чергою, призводить до проблеми зникнення градієнта, тоді NN перестає навчатися або навчається дуже повільно.

Функція гіперболічного тангенса (tanH activation function) схожа на функцію сигмоїда, але вона симетрична відносно початку координат. Це зумовлює різні знаки виходів з попередніх шарів, які будуть подаватися на вхід наступного шару. Її можна визначити за формулою 1.2.

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1. \qquad (1.2)$$

Функція гіперболічного тангенса є неперервною і диференційованою, її значення лежать в діапазоні від -1 до 1. У порівнянні з сигмоїдною функцією, градієнт функції є більш крутим. Дана функція є кращою за сигмоїдну функцію, оскільки вона має градієнти, які не обмежені у зміні в певному напрямку, а також має нульовий центр [18].

Графік даної функції зображено на рисунку 1.2.

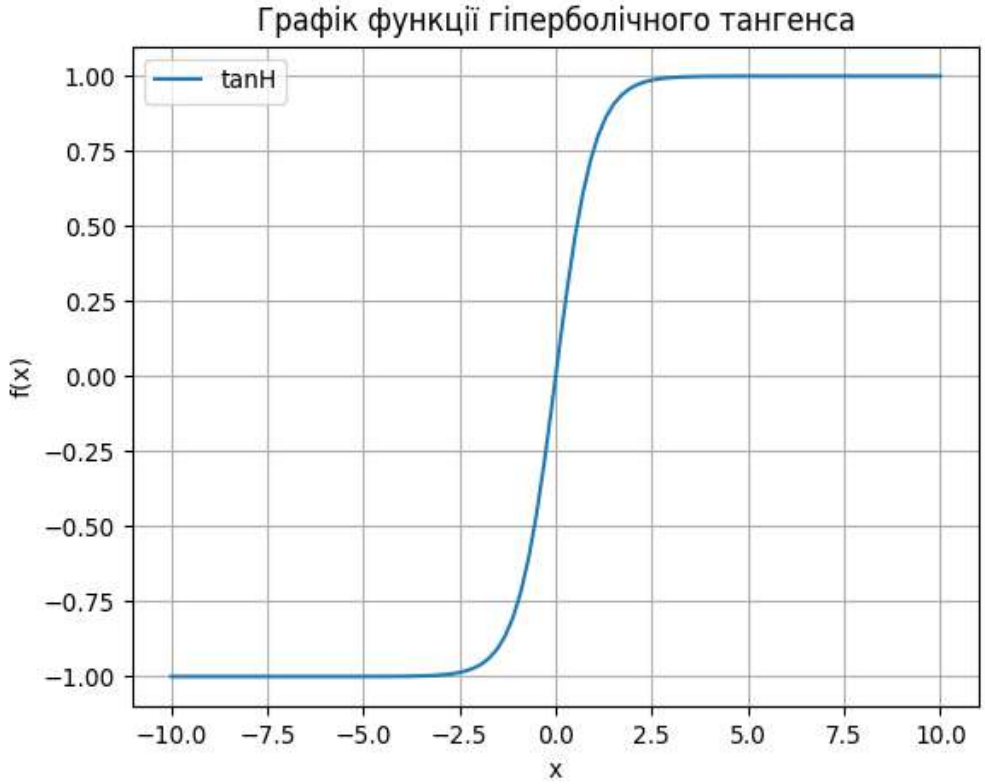


Рисунок 1.2 – Графік функції гіперболічного тангенса

Джерело: за даними [18]

Функція зрізаного лінійного вузла (rectified liner unit, далі – ReLU) є нелінійною FA, яка широко використовується в NNs.

Перевагою використання даної функції є те, що всі нейрони не активуються одночасно. Це означає, що нейрон буде деактивовано лише тоді, коли результат лінійного перетворення дорівнює нулю.

Математично це можна виразити за формулою 1.3.

$$f(x) = (0, x) \ = \{x, \text{for } x \geq 0 \ 0, \text{for } x < 0 \ . \qquad (1.3)$$

ReLU є більш ефективною, ніж інші функції, оскільки всі нейрони не активуються одночасно, а лише певна їх кількість. У деяких випадках значення градієнта дорівнює нулю, завдяки чому ваги та зсуви не оновлюються на кроці зворотного розповсюдження при навчанні NN [19].

Графік даної функції зображено на рисунку 1.3.

Рисунок 1.3 – Графік функція ReLU

Джерело: за даними [19]

Багатозмінна логістична функція (далі – softmax) – це комбінація декількох сигмоїдних функцій. Оскільки сигмоїдна функція повертає значення в діапазоні від 0 до 1, їх можна розглядати як ймовірності точок даних певного класу. Дана функція, на відміну від сигмоїдних, які використовуються для бінарної класифікації, може бути використана для задач багатокласової класифікації. Ця функція для кожної точки даних усіх окремих класів повертає ймовірність. Вона може бути виражена за формулою 1.4.

$$f(x)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \text{for } j = 1,2,3 \dots K. \quad (1.4)$$

При побудові мережі або моделі для багатокласової класифікації, вихідний шар мережі матиме таку ж кількість нейронів, як і кількість класів у цільовій вибірці [20].

Графік даної функції зображено на рисунку 1.4.

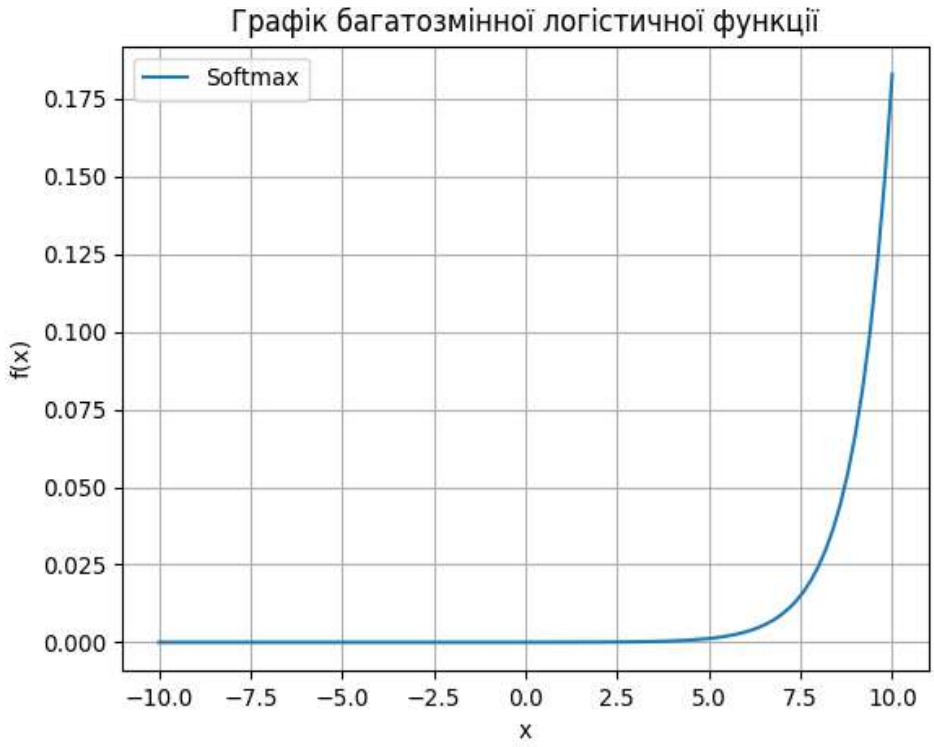


Рисунок 1.4 – Графік функції softmax

Джерело: за даними [20]

Отже, вибір відповідного шару активації залежить від призначення NN, типу даних і моделі. Для завдань класифікації, використовується багатозмінна логістична функція, якщо необхідно отримати ймовірності класів як вихідні дані та сигмоїдна або функція гіперболічного тангенса, якщо – двійковий вихід. Для моделей глибокого навчання, ReLU є загальним вибором для прихованих шарів, оскільки вона може прискорити навчання. Крім того, під час вибору FA необхідно враховувати властивості функції, як-от похідна, здатність забезпечувати нелінійність і здатність запобігати загасанню градієнта. Загалом, вибір відповідного шару активації залежить від конкретного завдання й експериментальних результатів. Необхідно ретельно підбирати FA й змінювати її, щоб отримати оптимальні результати для вашої моделі.

Функція втрат – це функція, яка приймає два значення – істинне і прогнозоване, і повертає оцінку порівняння, наскільки прогнозоване значення близьке до істинного. Чим менша ця оцінка, тим краще, і навпаки – великі значення оцінки вказують на те, що модель потребує доопрацювання. Основне завдання в процесі навчання NN, зводити до мінімуму різницю між істинним і прогнозованим значенням у готовій моделі [21]. Вибір функції втрат визначає спосіб, у який модель навчається робити прогнози на будь-якому наборі даних. Отже, дуже важливо вибрати оптимальну функцію втрат залежно від потреби, щоб отримати оптимальні результати.

Бінарна перехресна ентропія або логарифмічна втрата (binary cross entropy or logarithmic loss). При бінарній класифікації модель виводить значення від 0 до 1. Якщо це значення сильно відрізняється від очікуваного значення (яке дорівнює 0 або 1), то значення втрат буде високим, і модель буде сильно оштрафована. Якщо різниця між прогнозованим і очікуваним значеннями невелика, то витрата буде менше. Журнал втрати вказує на те, наскільки близька ймовірність передбачення до відповідного фактичного/справжнього значення (0 або 1 у випадку двійкової класифікації),

штрафуючи неточні передбачення вищими значеннями [22]. Менші логарифмічні втрати вказують на кращу продуктивність моделі (рис. 1.5).

Значення втрат розраховується за формулою 1.5.

$$\log\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p(y_i)\right) + (1 - y_i) \cdot \log\left(1 - p(y_i)\right), \quad (1.5)$$

де y_i – фактичний клас,

$\log\left(p(y_i)\right)$ – ймовірність його класу,

$p(y_i)$ – ймовірність 1,

$1 - p(y_i)$ – ймовірність 0.

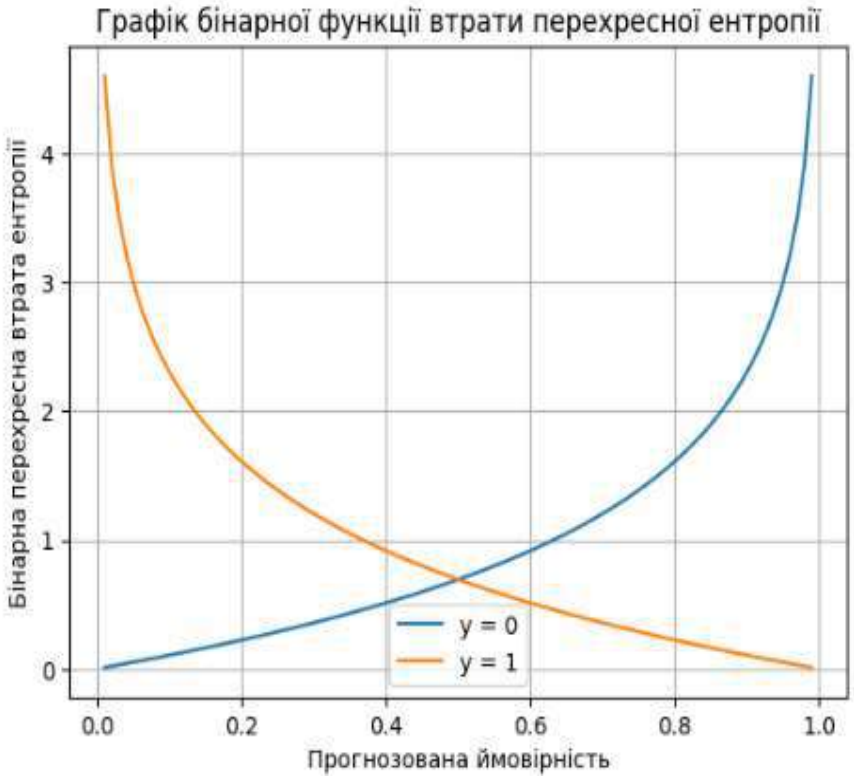


Рисунок 1.5 – Бінарна функція втрати перехресної ентропії

Джерело: за даними [22]

Категоріальна перехресна ентропія (categorical cross entropy or softmax loss) (рис. 1.6).

Це softmax поєднана з перехресною ентропійною втратою, яка використовується для багатокласової класифікації.

У багатокласовій класифікації вихідні дані NN проходять через активацію softmax, яка потім виводить вектор прогнозованих ймовірностей для вхідних класів.

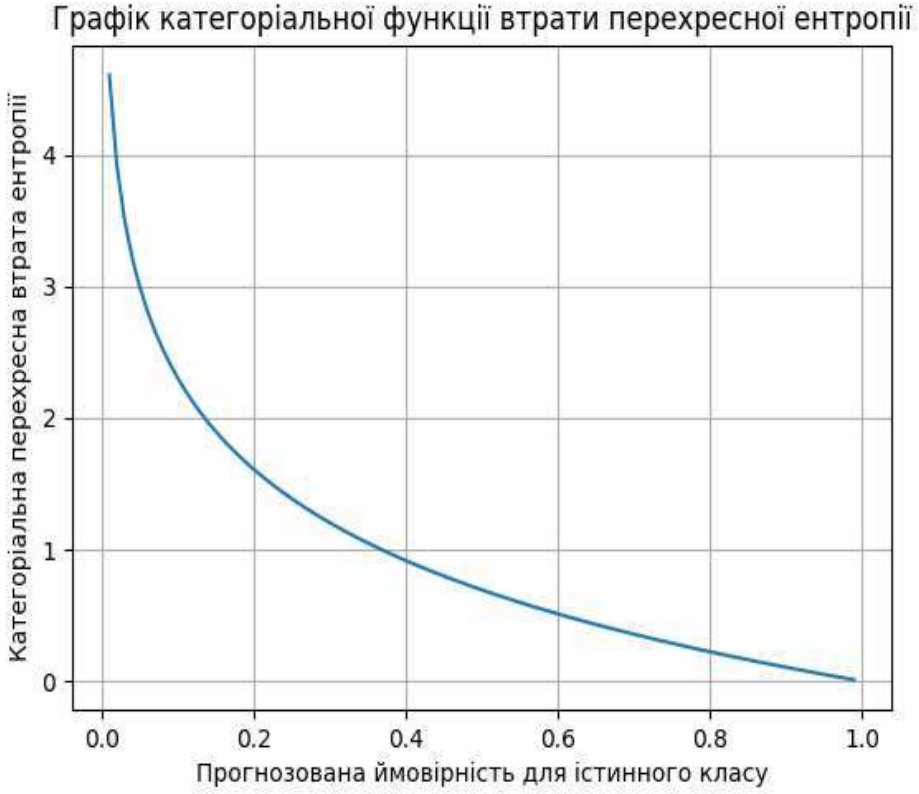


Рисунок 1.6 – Категоріальна перехресна ентропія

Джерело: за даними [22]

У багатокласовій класифікації результатом моделі буде вектор. Кількість елементів у векторі дорівнює кількості класів. Кожен елемент вихідного вектору представляє ймовірність того, що вхідна послідовність належить до певного класу. У випадку категоріальної перехресної ентропії очікуваний вектор матиме єдину 1, що означає, що вхідна послідовність натискань клавіш належить до відповідного класу, а всі інші елементи

дорівнюватимуть нулю. Після отримання очікуваного та передбаченого вектору, обчислюється наступне підсумовування для обчислення величини втрат (формула 1.6) [22].

$$\text{softmax loss} = -\sum_{i=1}^{N} y_i \cdot \log(y_i) . \qquad (1.6)$$

Основним завданням для якісного навчання моделі є визначення необхідної кількості епох.

Зупинка навчання раніше оптимальної кількості призводить до недостатньої пристосованості моделі. Якщо ж навчати модель протягом надмірно тривалого часу, то це призведе до перенавчання, через що модель буде погано працювати на реальних зовнішніх наборах даних.

Одним зі способів досягнення оптимального навчання моделі є використання концепції ранньої зупинки (early stopping). Вона полягає у відстеженні приросту валідаційних втрат після кожної епохи. Валідаційні дані відокремлюються від навчальних даних і можуть бути апроксимовані як узагальнений набір даних, що нагадує реальні дані [23].

У багатьох випадках важливо забезпечити якісну обробку та аналіз даних для використання в NNs. Це може включати предпроцесінг даних (data preprocessing), такий як масштабування, нормалізацію, видалення шуму або вибір певних ознак для подальшого використання. Крім того, для досягнення кращих результатів, можуть застосовуватись техніки оптимізації.

До таких технік належить застосування оптимізаторів, наприклад Адам. Оптимізатор Адама – це адаптивний алгоритм оптимізації швидкості навчання, який розроблено спеціально для навчання NNs. Це комбінація середньоквадратичного поширення (root mean square propagatio, далі – RMSprop) та стохастичного градієнтного спуску з імпульсом.

Імпульс використовується для прискорення алгоритму градієнтного спуску шляхом врахування «експоненціально зваженого середнього» градієнтів. Використання середніх значень змушує алгоритм наближатися до мінімумів швидше (формула 1.7) [24].

.

$$\omega_{t+1} = \omega_t - \alpha m_t\,, \tag{1.7}$$

де

$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta \omega_t}\right], \tag{1.8}$$

де m_t – сукупність градієнтів у момент часу t (поточний) (спочатку = 0),

m_{t-1} – сукупність градієнтів у момент часу $t-1$ (попередній),

ω_t – ваги в момент часу t,

ω_{t+1} – ваги в момент часу $t+1$,

α_t – швидкість навчання в момент часу t,

δL – похідна функції втрат,

$\delta \omega_t$ – похідна ваг у момент часу t,

β – параметр ковзного середнього (const, 0,9).

RMSprop – це адаптивний алгоритм навчання, який замість того, щоб брати кумулятивну суму квадратів градієнтів, бере «експоненціальне ковзне середнє» (формула 1.9) [25].

$$\omega_{t+1} = \omega_t - \frac{\alpha_t}{(v_t+\varepsilon)^{1/2}}\left[\frac{\delta L}{\delta \omega_t}\right], \tag{1.9}$$

де

$$v_t = \beta v_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta \omega_t}\right]^2, \tag{1.10}$$

де ω_t – ваги в момент часу t,

ω_{t+1} – ваги в момент часу $t + 1$,

α_t – швидкість навчання в момент часу t,

δL – похідна функції втрат,

$\delta\omega_t$ – похідна ваг у момент часу t,

v_y – сума квадратів минулі градієнти (спочатку = 0),

β – параметр ковзного середнього (const, 0,9),

ε – невелика позитивна константа (10^{-8}).

Оптимізатор Адама успадковує сильні сторони двох вищезазначених методів і ґрунтується на них, щоб забезпечити більш оптимізований градієнтний спуск.

З формул 1.7 та 1.9 отримуємо формулу 1.11 [25].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\left[\frac{\delta L}{\delta \omega_t}\right] v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left[\frac{\delta L}{\delta \omega_t}\right]^2 . \quad (1.11)$$

Оскільки обидва m_t і v_t ініціалізуються як 0, спостерігається тенденція до «зміщення до 0», оскільки β_1 і $\beta_2 \approx 1$. Цей оптимізатор розв'язує цю проблему за допомогою обчислення з корекцією зміщення m_t і v_t. Це також робиться для контролю ваги при досягненні глобального мінімуму, щоб запобігти сильним коливанням поблизу нього. Використовуючи формулу 1.12 [25].

$$\underline{m_t} = \frac{m_t}{1 - {\beta_1}^t} \; \underline{v_t} = \frac{v_t}{1 - {\beta_2}^t} \; . \quad (1.12)$$

Замість звичайних вагових параметрів m_t і v_t беруться вагові параметри з поправкою на зсув $\underline{m_t}$ t і $\underline{v_t}$. Помістивши їх у загальне рівняння 1.11, отримується формула 1.13 [25]:

$$\omega_{t+1} = \omega_t - \underline{m_t} \frac{\alpha}{\sqrt{\underline{v_t} + \varepsilon}}. \qquad (1.13)$$

Існує кілька типів архітектур, які використовуються для різних задач. Мережі прямого розповсюдження (feed-forward neural networks, далі – FNN) були винайдені найпершими та є найпростішими. Кожен нейрон у шарі приймає вхідні дані та генерує відповідний вихідний сигнал, який передається наступному шару (рис. 1.7).

Рекурентні NNs (recurrent neural networks, далі – RNN) часто використовуються для обробки послідовних даних, таких як мова або часові ряди.

На відміну від FNNs, які передають дані в одному напрямку та обробляють кожен шар тільки один раз, у RNNs присутня пам'ять. В RNN інформація проходить через мережу не тільки в одному напрямку, але й зворотньо.

Схематично порівняння FNN та RNN наведено на рисунку 1.8.

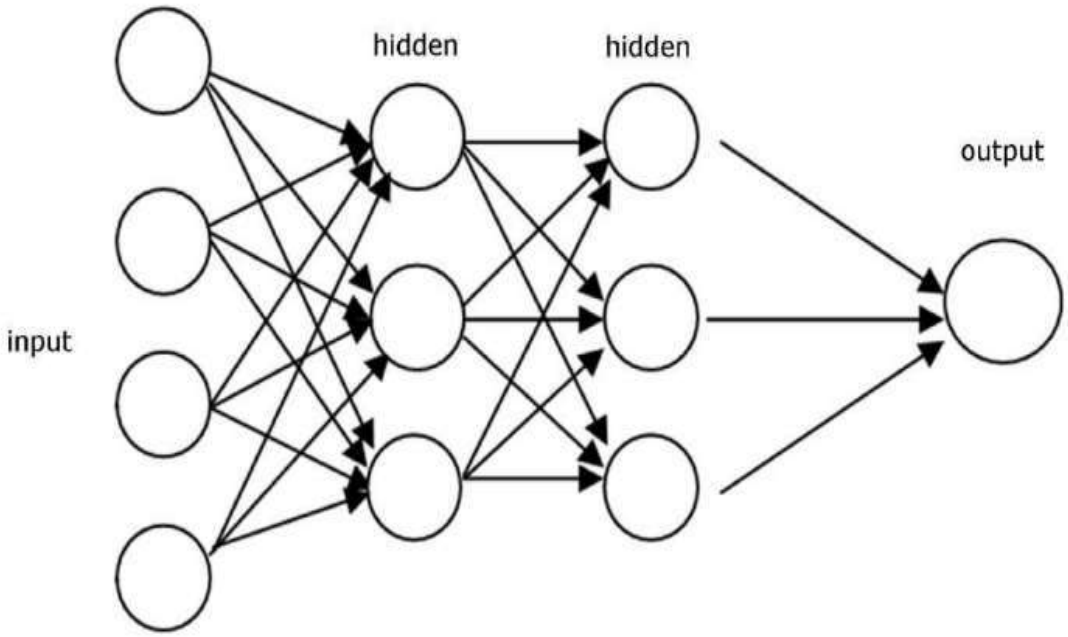
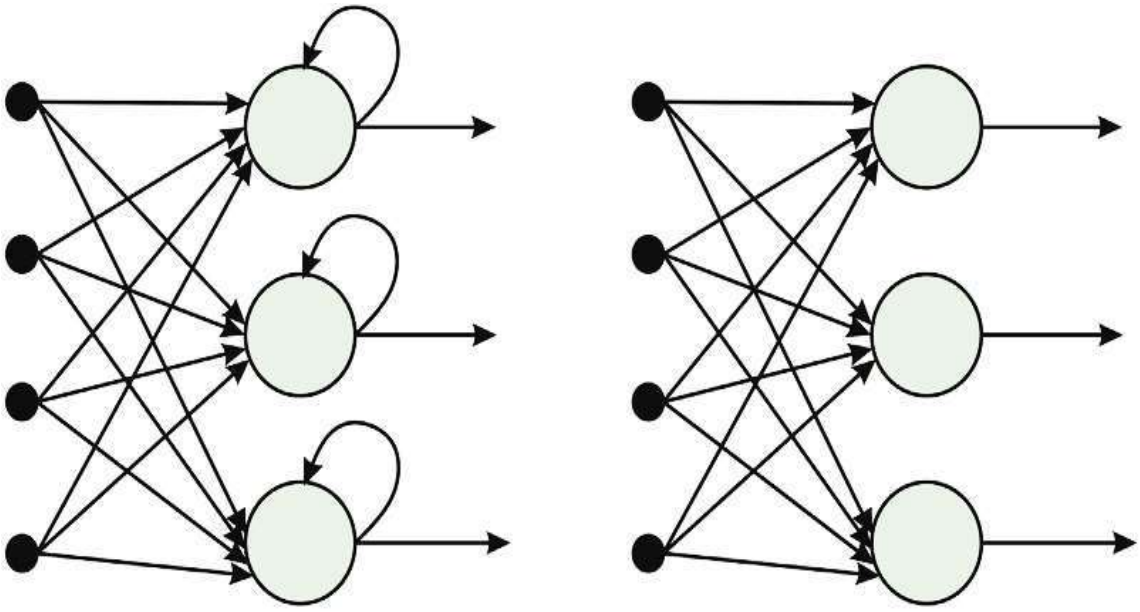


Рисунок 1.7 – Приклад FNN

Джерело: [26]



а) б)

Рисунок 1.8 – Приклад порівняння NN: а) RNN, б) FNN

Джерело: [27]

Це означає, що RNN здатна враховувати структуру послідовності, оскільки кожен наступний елемент послідовності враховує попередній контекст [28].

CNN показують високу точність при обробці зображень та використовуються у багатьох завданнях комп'ютерного зору.

CNN зазвичай складаються з послідовності шарів, які включають згорткові шари для витягування особливостей, шари підсумування (pooling layers) для зменшення розмірності та повнозв'язані шари (fully connected layers) для класифікації або регресії.

Загальною перевагою NNs є їх здатність до автоматичного вивчення складних залежностей та адаптації до різноманітних завдань. Однак, навчання NNs може бути вимогливим до обчислювальних ресурсів та часу, а також вимагати великого обсягу тренувальних даних для досягнення високої точності [29].

Для оцінки NNs використовуються деякі метрики, серед найбільш поширених є точність (accuracy), правильність (precision), фактичні

спостереження, що передбачені правильно (recall) та F1-міра. Ці метрики засновані на таких результатах: істинно позитивні (true positive, далі – TP) – фактичне та передбачене значення додатні, істинно негативні (true negative, далі – TN) – фактичне та передбачене значення негативні, хибнопозитивні (false positive, далі – FP) – фактичне значення негативне, але передбачене є позитивним, хибнонегативні (false negative, далі – FN) – фактичне значення є позитивним, але передбачене є негативним [30]. Хибнопозитивний і хибнонегативний результат ще називають помилками першого (false rejection rate, далі – FRR) і другого роду (false acceptance rate, далі – FAR) відповідно. Далі розберемо кожну з метрик окремо.

Accuracy – співвідношення між кількістю правильних передбачень (correct predictions) і загальною кількістю передбачень (total predictions), що знаходиться за формулою 1.14 [30]. Даний показник підходить для задач класифікації, які добре збалансовані та не мають перекосів, або не мають дисбалансу класів:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}. \quad (1.14)$$

Precision – відношення загальної кількості правильно класифікованих позитивних класів (predictions actually positive) до загальної кількості прогнозованих позитивних класів (total predicted positive), знаходиться за формулою 1.15 [30]. Даний показних є корисним у випадках, коли хибно-позитивний результат викликає більше занепокоєння, ніж хибнонегативний результат:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Predictions Actually Positive}}{\text{Total Predicted positive}}. \quad (1.15)$$

Recall – відношення загальної кількості правильно класифікованих позитивних класів (predictions actually positive) до загальної кількості позитивних класів (total actual positive), знаходиться за формулою 1.16 [30]. Даний показних є корисним у випадках, коли хибнонегативний результат переважає над хибнопозитивним.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}} \ . \qquad (1.16)$$

F1-міра – це число від 0 до 1, яке є середнім гармонійним значенням precision та recall. Дане значення використовується, тому що воно не чутливе до дуже великих значень, на відміну від простих середніх, знаходиться за формулою 1.17 [30].

$$\text{F1} - \text{Score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}. \qquad (1.17)$$

Ще одним інструментом, що використовується для оцінки точності мережі є побудова кривої помилок (receiver operating characteristic, далі – ROC-крива) та площі під цією кривою (area under the roc curve, далі – AUC), де вісь X – частота хибнопозитивних результатів (false positive rate, далі – FPR) є відношенням хибнопозитивних прогнозів (помилково класифікованих як позитивні) до загальної кількості фактичних негативних результатів, що обчислюється за формулою 1.18.

$$\text{Частота FP} = \frac{\text{FP}}{\text{FP} + \text{TN}} \ . \qquad (1.18)$$

Вісь Y – частота істино позитивних результатів (recall) розраховується за формулою 1.16.

AUC це окреме числове значення, яке кількісно оцінює загальну якість класифікатора. Значення AUC може знаходиться в діапазоні від 0 до 1, де при $\mathrm{AUC} = 1$ можна вважати класифікатор ідеальним, а $\mathrm{AUC} = 0{,}5$ означає, що він працює навмання.

Також для оцінювання використовують матрицю плутанини (confusion matrix). Матриця плутанини – це матриця NxN, яка використовується для оцінки якості моделі класифікації, де N – кількість класів. Матриця порівнює фактичні значення з тими, що були передбачені моделлю.

В контексті аутентифікації, NNs можуть бути навчені розпізнавати характеристики користувачів, такі як голос, обличчя, рукописний підпис, KD тощо. При цьому, NNs можуть дуже точно ідентифікувати конкретного користувача на основі його унікальних характеристик, що забезпечує високий рівень безпеки та захисту від несанкціонованого доступу до ресурсів. У випадку аутентифікації за KD NNs навчають на прикладах KD різних користувачів.

Для цього потрібно мати набір даних, який містить інформацію про KD кожного користувача. Набір даних може містити інформацію про час натискання та часу між натисканням кожної клавіші. Крім того, можна використовувати інші параметри, такі як відстань між клавішами, швидкість набору тексту тощо.

Після того, як дані з клавіатури користувачів будуть зібрані, використовуються NNs для створення моделей, які можуть ідентифікувати користувача на основі його KD.

1.3.2 Архітектури нейронних мереж, що використовуються в завданні аутентифікації за клавіатурним почерком

Для завдання аутентифікації за KD використовуються різні архітектури NNs. Одна з архітектур, яка використовується в завданні аутентифікації за KD, – це RNN [31-32].

RNNs – це клас NNs, які обробляють послідовності даних. Вони використовують зворотний зв'язок для передачі інформації між тимчасовими кроками, що дозволяє їм обробляти послідовності даних, такі як мовлення чи текст.

В аутентифікації за KD RNN може бути використана для моделювання послідовностей натискання клавіш та відступів між ними. Кожен нейрон у RNN зберігає внутрішній стан, що дозволяє передавати інформацію з попередніх часових кроків до наступних. Це дозволяє RNN враховувати контекстуальні залежності в послідовності клавіш та розпізнавати унікальні особливості клавіатурного почерку користувача.

Для завдання аутентифікації може використовуватися LSTM або керовані рекурентні блоки (gated recurrent unit, далі – GRU), які можуть моделювати залежності між послідовними подіями в KD. LSTM і GRU були створені як рішення для короткочасної пам'яті.

Вони мають внутрішні механізми, які називаються вентилями, що можуть регулювати потік інформації. Ці вентилі можуть визначати, які дані в послідовності важливо зберегти, а які – відкинути.

Таким чином, вони можуть передавати відповідну інформацію вниз по довгому ланцюжку послідовностей, щоб робити прогнози. Майже всі найсучасніші результати, засновані на RNNs, досягаються за допомогою цих двох мереж. LSTM і GRU можна знайти в розпізнаванні мови, синтезі мови та генерації тексту (рис. 1.9) [31].

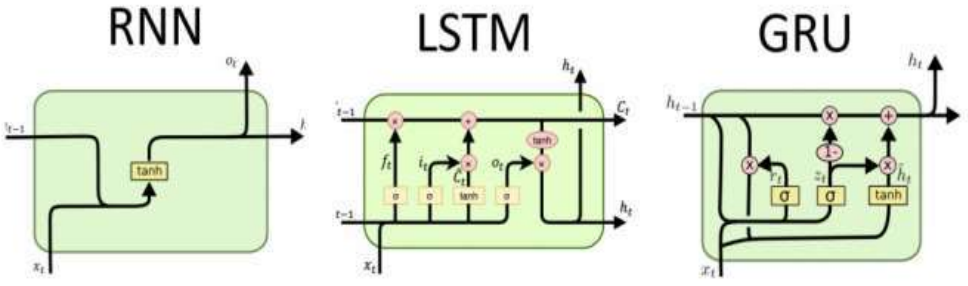


Рисунок 1.9 – Ядро LSTM, GRU та простої RNN

Джерело: [31]

В роботі [32] наведено підхід до аутентифікації користувача за допомогою динаміки натискання клавіш. LSTM-мережа складається з чотирьох шарів (рис. 1.10): вхідного шару, LSTM-шару, пов'язаного шару та вихідного шару.

Вхідний рівень отримує характеристики динаміки натискання клавіш, витягнуті алгоритмом, а рівень LSTM обробляє вхідну послідовність і вивчає шаблони поведінки кожного користувача при наборі тексту. Повністю з'єднаний шар і вихідний шар класифікують користувача як справжнього або самозванця.

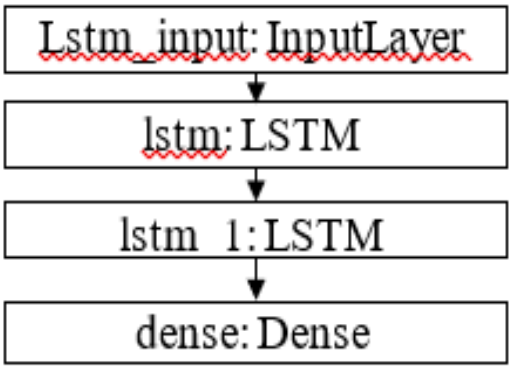


Рисунок 1.10 – Візуальне уявлення моделі

Джерело: [32]

Загалом, демонструється 80% точності при досить великій кількості епох навчання.

В [33] пропонується нова модель NNs для аутентифікації динаміки натискання клавіш на мобільних пристроях.

Запропонована модель складається з двох частин: модуля виділення ознак і модуля класифікації. Модуль вилучення функцій витягує різні статистичні та часові характеристики з поведінки користувача при наборі тексту, включаючи час утримання, час польоту та затримку натискання клавіш. Модуль класифікації використовує архітектуру NNs, зокрема LSTM, щоб класифікувати користувачів на основі їх шаблонів натискання клавіш.

Ця система досягла помилки другого роду та помилки першого роду на рівні 4,4% та 5,3% відповідно.

Загалом у комірці LSTM є додаткові вентилі, а саме вхідні, забування та вихідні, які використовуються для того, щоб вирішити, які сигнали будуть передані до іншого вузла. Рівняння 1.19 – 1.24 описують поведінку всіх вентилів у комірці LSTM, ядро наведено на рисунку 1.11 [34].

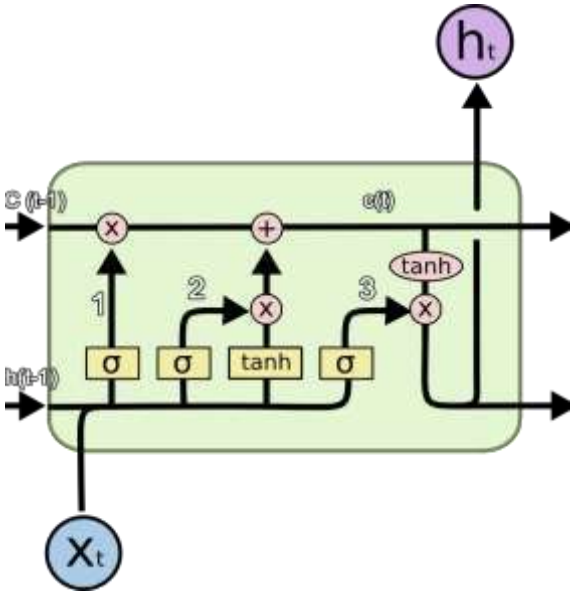


Рисунок 1.11 – Ядро LSTM

Джерело: [34]

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right), \tag{1.19}$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right), \tag{1.20}$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o), \tag{1.21}$$

$$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g), \tag{1.22}$$

$$C_t = \sigma\left(f_t C_{t-1} + i_t \tilde{C}_t\right), \tag{1.23}$$

$$h_t = \tanh(C_t) o_t), \tag{1.24}$$

де W – рекурентний зв'язок між попереднім прихованим шаром і поточним прихованим шаром,

U – вагова матриця, яка з'єднує входи з прихованим шаром,

$\tilde{C}$ – потенційний прихований стан, який обчислюється на основі поточного входу та попереднього прихованого стану,

C – внутрішня пам'ять пристрою, яка є комбінацією попередньої пам'яті, помноженої на вентиль забування, та нового обчисленого прихованого стану, помноженого на вхідний вентиль [34].

Також використовується й інші архітектури NNs, що використовуються в завданні аутентифікації за KD, наприклад CNN.

Використання згорткових шарів дозволяє витягувати просторові шаблони та особливості з введеного клавіатурного почерку. CNN може розпізнавати шаблони клавіш та їх послідовності, що відображають унікальний стиль письма кожного користувача [35].

В [36] пропонується метод аутентифікації користувачів, який ґрунтується на використанні CNN для аналізу KD в довільному тексті. Для цього розроблена процедура, яка перетворює довільний текст у формат квадратного багатоканального кольорового зображення.

Результати експериментів підтверджують, що запропонований метод нейромережевої аутентифікації демонструє точність розпізнавання користувачів на 10% вище, ніж точність розпізнавання, отримана за допомогою методу опорних векторів. Крім того, в дослідженні була виявлена необхідність дослідження перспективних типів CNNs для подальшого вдосконалення цього підходу.

Аналогічні результати також були отримані в [37-38], які головним чином зосереджені на застосування CNN типу LeNet для підвищення надійності систем біометричної аутентифікації користувачів на основі аналізу KD. Значення FRR становить 13% та 6%, коли структуру навчальних даних скореговано таким чином, щоб зробити модель CNN менш чутливою до легальних користувачів.

Застосування CNN до аутентифікації за KD може включати перетворення характеристик клавіш у двовимірне представлення та використання шарів згортки для виявлення локальних шаблонів та особливостей KD, для цього застосовуються 2-вимірні CNNs, тобто CNNs ядро, яких ковзає вздовж 2-вимірних даних.

1D CNN застосовується, коли необхідно отримати важливі ознаки з коротких сегментів загального набору даних і коли розташування ознаки в межах сегмента не має великого значення. Це підходить для аналізу часових послідовностей. Іншим застосуванням є обробка природної мови, хоча сьогодні існують кращі моделі для цього [39].

Ключова відмінність полягає в розмірах вхідних даних і в тому, як детектор ознак (або фільтр) їх аналізує в процесі обробки. Як показано на рисунку 1.13, ядро одномірного CNN рухається у вертикальному напрямку, тоді як двомірний CNN покроково переглядає тривимірні вхідні дані не лише по вертикалі, але й по горизонталі [40].

Форма згорткового фільтра (ядра) – це векторна форма в 1D CNN і, як правило, матрична форма у 2D CNN, як показано на рис. 1.12.

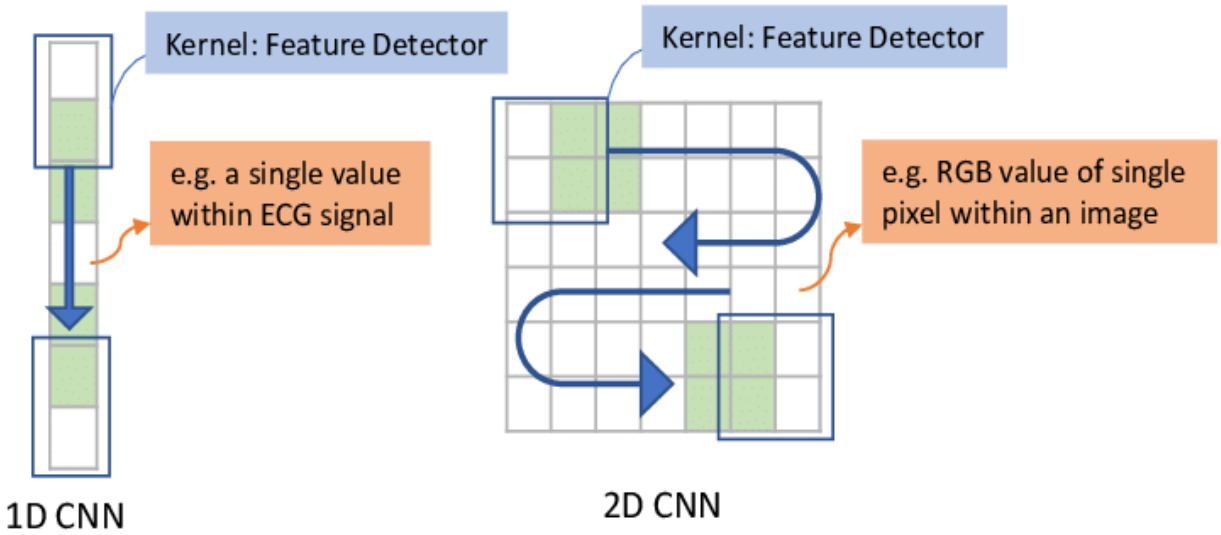


Рисунок 1.12 – Рух ядра на 1D CNN та 2D CNN

Джерело: [39]

Очевидно, що для підвищення рівня безпеки при аутентифікації користувачів постає потреба об'єднання декількох видів NNs.

В [40] наводиться приклад поєднання RNNs та CNNs. Запропонований метод передбачає збір даних динаміки натискання клавіш від користувачів, коли вони вводять довільний текст, який потім попередньо обробляється для вилучення параметрів. Витягнуті функції потім використовуються як вхідні дані для CNNs, які навчаються класифікувати функції як належні користувачеві чи самозванцю. Вихідні дані CNNs потім подаються в RNNs, які вчаться моделювати часові залежності даних натискання клавіш. Вихідні дані RNNs використовуються як остаточне рішення аутентифікації.

Запропонований метод безперервної аутентифікації за допомогою вільного тексту з використанням CNNs і RNNs ефективний для аутентифікації користувачів і виявлення самозванців. Модель була протестована з використанням відкритого набору даних, і найкращий показник помилкових відмов (FRR) склав 1,95%, показник помилкових прийняттів (FAR) – 4,12%, а показник рівних помилок (equal error rate, EER) – 3,04% [40].

1.4 Визначення мети та основних задач роботи

В результаті всього вищесказаного сформована наступна мета магістерської роботи: підвищення рівня захисту комп’ютерно-інформаційних систем шляхом високоточної автоматичної ідентифікації користувачів за їх клавіатурним почерком за допомогою нейронної мережі аутентифікації, що забезпечить ідентифікацію користувача на основі унікальних характеристик його введення на клавіатурі.

Для досягнення поставленої мети сформульовані наступні задачі:

- на основі аналізу сучасних методів та засобів ідентифікації користувачів обрати архітектуру NN, застосування якої забезпечує високу точність в завданні аутентифікації за KD;

- запропонувати архітектуру NN з використанням CNN та LSTM шарів, які можуть враховувати послідовності натискання клавіш та моделювати унікальний стиль KD кожного користувача;
- провести тренування та налаштування NN на наборі тренувальних даних, використовуючи відповідні методи оптимізації, щоб досягнути максимальної точності моделі;
- оцінити якість розробленої моделі на наборі тестових даних, для оцінки її здатності розрізняти аутентичний почерк від підробленого.

1.5 Висновок за розділом

У даному розділі були оглянуті різні методи аутентифікації користувача за KD. Статистичні методи використовують статистичні характеристики KD, динамічні підходи враховують динамічну інформацію про послідовності натискання клавіш, а HMMs можуть моделювати послідовності натискання клавіш. Також розглянуто використання NNs, зокрема CNNs, RNNs та їх комбінації.

Застосування аутентифікації за KD має свої переваги, такі як висока унікальність індивідуального почерку, зручність використання і низькі витрати. Використання статистичних методів і динамічних підходів може забезпечити задовільну точність аутентифікації, а HMM дозволяє моделювати складність KD.

Проте, існують деякі проблеми, пов'язані з застосуванням NNs для аутентифікації за KD. Наприклад, у деяких випадках можуть виникати проблеми зі збором достатньої кількості даних для навчання мережі, а також зі зменшенням точності при роботі з великими об'ємами даних.

Загалом, аутентифікація користувача за KD є перспективним напрямком досліджень та має потенціал стати якісним інструментом для забезпечення безпеки та ідентифікації користувачів.

Наприкінці на основі проведеного аналізу визначені задачі, яким буде присвячена магістерська робота.

РОЗДІЛ 2
АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ АУТЕНТИФІКАЦІЇ КОРИСТУВАЧА ЗА КЛАВІАТУРНИМ ПОЧЕРКОМ

Враховуючи вище викладене пропонується модифікована архітектура NN для аутентифікації за KD. Дана мережа заснована на поєднанні CNN та LSTM. Порівняно з описаними в першому розділі NNs для аутентифікації за KD, вона очікувано може мати високу точність аутентифікації користувача за KD та низький коефіцієнт помилкового розпізнавання.

Пропонована архітектура наведена на рисунку 2.1.

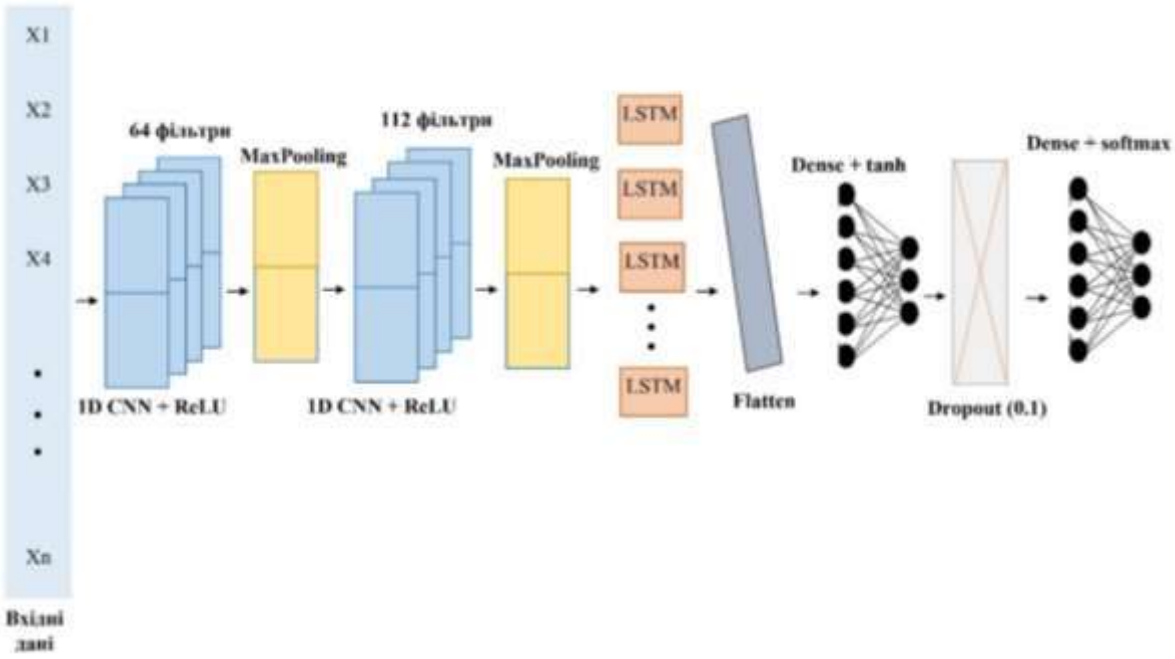


Рисунок 2.1 – Архітектура мережі

Джерело: авторська розробка

Ця модель аналізує послідовності даних, використовуючи згорткові шари та LSTM для виявлення важливих ознак. Вона починає з одновимірних згорткових шарів з ReLU активацією та після них – шари максимального пулінгу для виділення ключових ознак.

Потім використовується LSTM для розуміння послідовностей та довгострокових залежностей. Після LSTM модель використовує шари Flatten, щоб перетворити дані в плоский тензор, та повнозв'язний шар з функцією активації гіперболічного тангенса.

Для регуляризації використовується шар Dropout, який випадковим чином вимикає деякі нейрони, та завершується Dense шаром з функцією активації softmax для передбачення класу вхідних прикладів, які відповідають кількості класів, які необхідно класифікувати (наприклад, користувачі).

Опис кожного шару, що наявний в архітектурі, буде описаний далі, але для початку вкажемо засоби, що використано при побудові моделі.

2.1 Вибір середи розробки

Ця модель розроблена мовою програмування Python, яка є найбільш поширеною для вирішення завдань машинного навчання, обробки та класифікації даних. У Python існує велика кількість бібліотек, які допомагають створити якісний продукт. Для розробки цієї системи використовувалася середа програмування Google Colaboratory. В основі Colaboratory лежить блокнот Jupyter для роботи на Python, тільки з базою на Google Диску, а не на комп'ютері. Це означає, що є можливість писати програми на Python без необхідності завантажувати бібліотеки та перевантажувати жорсткий диск власного комп'ютера, за умови наявності облікового запису Google [41].

Ця платформа дозволяє виконувати окремі частини коду, що спрощує роботу. Сервіс надає безплатний доступ до графічного процесора, але з обмеженою пам'яттю.

Основною перевагою є наявність безоплатних потужних графічних процесорів GPU (graphics processing unit) та TPU (google tensor processing unit), що дає можливість займатися не лише базовою аналітикою даних, а також проводити більш складні дослідження в галузі машинного навчання. Те, що центральний процесор (central processing unit, CPU) робить годинами, графічні процесори виконують за кілька хвилин або секунди [41].

Основні бібліотеки, використані при створенні цієї мережі, включають matplotlib, numpy, Keras та TensorFlow. Ця середа була обрана через її велику потужність, яка перевищує можливості стандартного комп'ютера.

2.2 Структура моделі

Структура запропонованої архітектури NN наведено на рисунку 2.2.

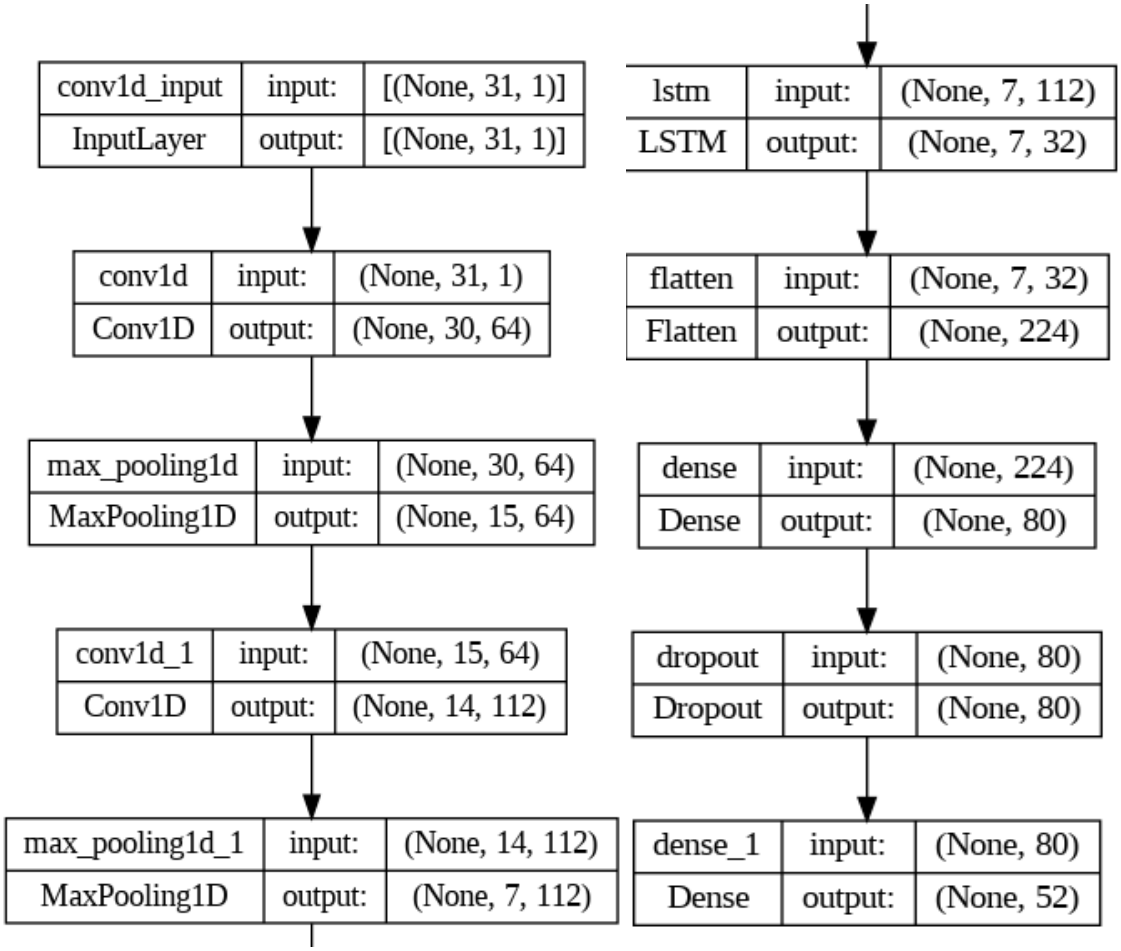


Рисунок 2.2 – Структура запропонованої моделі

Джерело: авторська розробка

Першим рівнем у цій моделі є шар Conv1D, що являє собою одновимірне згорткове ядро (1D convolutional layer), яке використовують для вилучення ознак із вхідних даних. Саме одновимірна згортка зазвичай використовується для даних послідовностей, таких як часові ряди, на відміну від 2D convolutional layer, який працює з двовимірними зображеннями. Даний рівень створено з наступними параметрами.

Filters – кількість вхідних фільтрів в згортці, що застосовуються на цьому шарі. Кожне ядро використовується для вилучення різних ознак із вхідних даних. Збільшення кількості фільтрів також збільшує складність моделі, що може призвести до переобладнання, якщо не керувати належним чином. В даній моделі використано значення 64.

Kernel_size – довжина вікна згортки 1D. Оскільки менший розмір ядра означає, що рівень охоплюватиме дрібніші деталі даних, тоді як більший розмір ядра охоплюватиме ширші та загальніші характеристики, то в даній моделі використано значення 2, це означає, що згортання виконується над групою двох сусідніх елементів вхідної послідовності.

Activation – FA, що буде використовуватись, її опис наведено в попередньому розділі даної роботи. В даній моделі обрано функцію ReLU.

Код додавання даного шару наведено на рисунку 2.3.

```
model1.add(tf.keras.layers.Conv1D(64,2, activation='relu', input_shape=(X_train.shape[1], 1)))
```

Рисунок 2.3 – Додавання шару Conv1D

Джерело: авторська розробка

Значення None на рисунку 2.2, означає, що розмір пакета даних буде визначено під час навчання, оскільки він може варіюватися залежно від даних, що подаються на вхід моделі. Загалом даний шар виконує операцію згортання на вхідних даних за допомогою ядра заданого розміру і ваг, що налаштовуються, що дає змогу моделі виявляти локальні ознаки і структури в даних. Вихідні дані цього шару будуть передаватися далі в наступні шари для подальшого опрацювання.

Наступний шар MaxPooling1D у даній моделі є одновимірним шаром максимального пулінгу (max-pooling layer), який використовується для зменшення розмірності даних і виділення найважливіших ознак. Основним параметром MaxPooling1D шару є pool_size – розмір вікна, яке переміщується

по вхідних даних. В даній роботі моделі він дорівнює 2, тобто для кожних двох послідовних елементів даних обирається максимальне значення.

Код додавання даного шару наведено на рисунку 2.4.

```
model1.add(tf.keras.layers.MaxPooling1D(pool_size=2))
```

Рисунок 2.4 – Додавання шару MaxPooling1D

Джерело: авторська розробка

Цей шар розбиває вхідні дані на непересічні вікна заданого розміру 2 X 2. Для кожного вікна вибирає найбільший елемент і передає його у вихідну послідовність. Таким чином, вихідні дані містять найбільші значення з кожної групи значень вхідної послідовності, що дає змогу зберегти найважливіші ознаки та зменшити розмірність даних.

На виході даного шару після операції максимального пулінгу отримано, що довжину вихідної послідовності – 15 та кількість фільтрів без змін – 64.

Далі йдуть послідовно ще два шари Conv1D та MaxPooling1D. Завдяки чому, повторно виконується згортка з 112 фільтрами і ядром розміром 2, що дає змогу витягти більш високорівневі ознаки.

Код додавання наведено на рисунку 2.5.

```
model1.add(tf.keras.layers.Conv1D(112,2, activation='relu'))
model1.add(tf.keras.layers.MaxPooling1D(pool_size=2))
```

Рисунок 2.5 – Додавання шарів Conv1D та MaxPooling1D

Джерело: авторська розробка

Наступним, у запропонованій моделі йде LSTM, що є шаром довгої короткострокової пам'яті і використовується для моделювання послідовних даних і вловлювання довгострокових залежностей у вхідних даних. Шар

складається з 32 прихованих нейронів (блоків пам’яті). Кожен прихований нейрон вловлює різні тимчасові залежності і структури у вхідних даних. Параметр return_sequences=True вказує, що цей шар повинен повертати послідовність виводу, а не тільки останній вихідний елемент.

Код додавання даного шару наведено на рисунку 2.6.

```
model1.add(tf.keras.layers.LSTM(32, return_sequences=True))
```

Рисунок 2.6 – Додавання шару LSTM

Джерело: авторська розробка

Наступний шар виконує операцію вирівнювання даних перед передачею їх у повнозв'язний шар. Цей шар не має параметрів для навчання і просто змінює форму вхідних даних, перетворюючи багатовимірний тензор на одновимірний.

У даній моделі після операцій згортання і пулінгу, отримується вихідний тензор із тривимірною формою, де один вимір відповідатиме за розмір пакета (batch size), а два інших виміри відповідатимуть за висоту і ширину даних. Однак перед подачею даних у повнозв'язний шар Dense необхідно перетворити цей тривимірний тензор на одновимірний, щоб кожен елемент даних став доступним для опрацювання повнозв'язним шаром.

Код додавання даного шару наведено на рисунку 2.7.

```
model1.add(tf.keras.layers.Flatten())
```

Рисунок 2.7 – Додавання шару Flatten()

Джерело: авторська розробка

Шар Dense є повнозв'язним шаром та складається зі 80 нейронів, і кожен нейрон з'єднаний з кожним нейроном попереднього шару (у цьому випадку, з

виходом шару Flatten()). FA визначає, як нейрон реагує на вхідні дані і які значення він генерує на виході. Активація Tanh трансформує вхідні значення в діапазоні від -1 до 1, роблячи його ненасичуваною функцією, яка може набувати від'ємних і позитивних значень.

Код додавання даного шару наведено на рисунку 2.8.

```
model1.add(tf.keras.layers.Dense(80, activation='tanh'))
```

Рисунок 2.8 – Додавання шару Dense

Джерело: авторська розробка

Шар Dropout виконує регуляризацію моделі шляхом випадкового вимкнення (обнулення) частини нейронів їхніх вихідних значень у процесі навчання, що дозволяє зменшити перенавчання моделі.

Шар Dropout не має параметрів, що навчаються, але в нього є один важливий гіперпараметр, званий «коефіцієнт відсіву» (dropout rate). У цьому разі ймовірність вимкнення нейрона встановлено рівною 0.1, що означає, що в середньому 10% нейронів буде вимкнена на кожному навчальному кроці.

Код додавання даного шару наведено на рисунку 2.9.

```
model1.add(Dropout(0.1))
```

Рисунок 2.9 – Додавання шару Dropout

Джерело: авторська розробка

Останнім шаром є Dense, що складається з unique_cols нейронів, де unique_cols являє собою кількість категорій, на які необхідно класифікувати дані. FA softmax використовується для виконання багатокласової класифікації.

FA softmax перетворює вихідні значення нейронів на ймовірності, причому сума ймовірностей для всіх класів дорівнює 1. Таким чином, цей шар

генерує ймовірнісний розподіл для кожного класу, даючи змогу моделі передбачати, до якого класу належить вхідний приклад.

Після навчання цієї моделі, вона буде здатна виконувати багатокласову класифікацію на unique_cols різних класів. Вихід цього шару міститиме ймовірності для кожного класу, і клас із найбільшою ймовірністю вважатиметься передбаченням моделі для даного вхідного прикладу.

Код додавання даного шару наведено на рисунку 2.10.

```
model1.add(tf.keras.layers.Dense(units=unique_cols,activation='softmax'))
```

Рисунок 2.10 – Додавання шару Dense

Джерело: авторська розробка

Далі модель скомпіловано (рис. 2.11), тобто перевірено наявність помилок формату, визначено функцію втрат та оптимізатор. Скомпільовану моделі надалі використано для навчання.

```
model1.compile(optimizer= 'adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Рисунок 2.11 – Компілювання моделі

Джерело: авторська розробка

В 4 розділі даної роботи буде описано, чому обрано саме такі параметри запропонованої моделі, виконано навчання та оцінювання отриманої моделі. В наступному розділі описано дані, які використовувались для навчання моделі.

2.3 Висновок за розділом

У даному розділі була представлена архітектура NN для аутентифікації користувача за допомогою KD. Модель була розроблена з використанням

мови програмування Python та низки популярних бібліотек, таких як matplotlib, numpy, Keras та TensorFlow.

Архітектура мережі поєднує в собі згорткові (Conv1D) та рекурентні (LSTM) шари, спрямовані на аналіз послідовних даних KD. Ця комбінація дозволяє виявляти залежності у вхідних даних, що робить модель якісною в завданні аутентифікації користувача.

Детальний опис кожного шару, включаючи згорткові, пулінгові, LSTM, та повнозв'язні, надає чітку картину того, як модель обробляє дані та виконує класифікацію. Важливим етапом є використання функції активації та методу викидання (dropout), що сприяє регуляризації моделі та запобігає перенавчанню.

Загальна архітектура мережі потенційно дозволить досягти високої точності аутентифікації користувача за KD, при цьому забезпечуючи низький рівень помилкового розпізнавання.

РОЗДІЛ 3
ПІДГОТОВКА ДАНИХ ДЛЯ НАВЧАННЯ МЕРЕЖІ АУТЕНТИФІКАЦІЇ КОРИСТУВАЧА ЗА КЛАВІАТУРНИМ ПОЧЕРКОМ

Як вхідні дані для навчання NN для аутентифікації за KD потрібні великі набори даних KD різних користувачів. Створення таких наборів даних з нуля потребує багато часу та витрат, тим більше, що для того, щоб дійсно добре навчити NN, набір повинен бути дуже великим, різноманітним, а також збалансованим за всіма параметрами KD. Тому зазвичай використовують вже готові дані, на яких навчають, тестують і оцінюють моделі NNs.

Всі ці набори даних знаходяться в публічному доступі, деякі з них є умовними стандартами, на яких часто порівнюють існуючі моделі NNs один з одним для отримання більшої точності, оскільки якість аутентифікації сильно залежить від вхідного набору даних і не зовсім правильно порівнювати NNs, навчені на різних наборах даних.

Існує значна кількість наборів даних для різних цілей, а також набори, створені спеціально для тестування моделей аутентифікації користувача за KD. Як вхідні дані для навчання NN для аутентифікації користувача обрано датасет [42], що був зібраний у Carnegie Mellon University для випробування різних моделей розпізнавання та доданий до нього авторську спробу вводу.

3.1 Опис процесу збору даних клавіатурного введення

Дані складаються з інформації про час натискання клавіш від п'ятдесяти одного користувача, кожен з яких вводить пароль .tie5Roanl 400 разів. У [43] зазначено, що пароль був згенерований випадково і за правилами складання паролів є складним, бо містить розділові знаки, цифри, великі та маленькі літери.

Для збору даних встановлено ноутбук із зовнішньою клавіатурою та розроблений додаток для Windows, який пропонує досліджуваному ввести

пароль. Програма відображає пароль на екрані з полем для введення тексту. Для того, щоб перейти до наступного екрана, користувач повинен правильно, послідовно ввести 10 символів пароля, а потім натиснути клавішу Enter. Якщо в послідовності будуть виявлені помилки, йому буде запропоновано повторити введення пароля. Для завершення сеансу збору даних необхідно правильно ввести пароль 50 разів [43].

Під час створення датасету Carnegie Mellon University [42] викликано випадкових п’ятдесят одну людину, кожен з яких протягом 8 сесій (кожна сесія проходила не частіше ніж раз на день) набирав цей пароль 50 разів, що дозволило забезпечити точність часових міток та максимально однакові умови, оскільки якби деякі користувачі вводили пароль частіше, ніж інші, або якби вони використовували різні клавіатури, ці відмінності могли б призвести до спотворення даних та штучно полегшити подальше розпізнавання користувачів.

Цей датасет обрано для аналізу інформативності часових ознак KD в першу чергу через його відкритість, а, по-друге, через рівномірність та повноту дослідних даних.

Помилки введення не приймалися, тобто при помилці необхідно перенабирати пароль заново. Дані зібрані дуже грамотно, розподілені в часі й дозволяють звикнути до пароля, що імітує ситуацію, коли пароль вводиться на автоматі «м'язовою пам'яттю».

Отже, оскільки пароль один і всі користувачі бачили його вперше і набирали в одних і тих же умовах, єдиною їх відмінністю має стати саме динаміка натискання клавіатури.

Необроблені записи всіх натискань клавіш і часових позначок суб’єктів були проаналізовані, щоб створити таблицю синхронізації паролів. Таблиця синхронізації паролів кодує характеристики часу для кожного з 400 паролів, які кожен суб’єкт ввів.

Також до датасету Carnegie Mellon University додано авторські спроби вводу. Той же пароль вводився також протягом 8 сесій по 50 разів у кожній, загалом 400 разів, автору присвоєно ідентифікатор s001.

Для збору авторських даних була використана створена програма. На головному екрані виводиться вікно, в якому потрібно ввести пароль (рис.3.1).

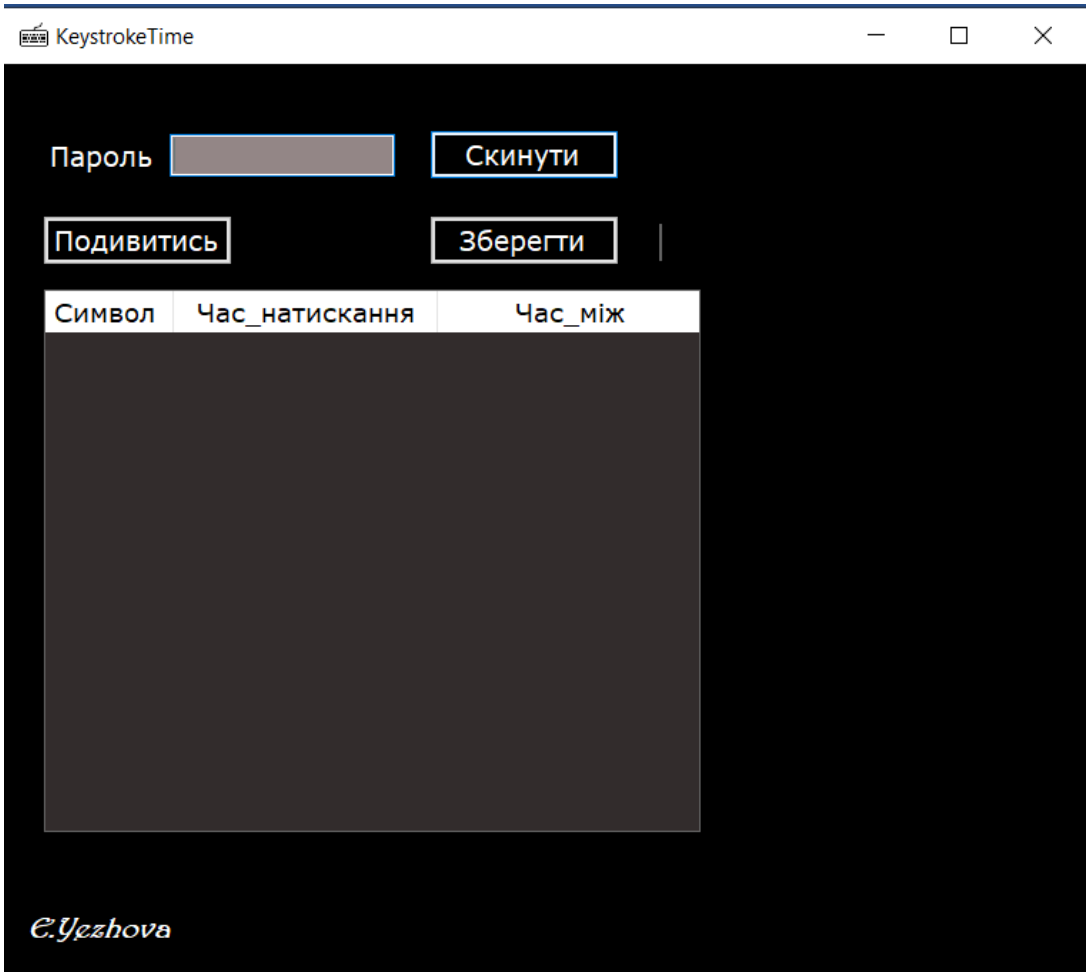


Рисунок 3.1 – Головне вікно

Джерело: авторська розробка

Якщо користувач натисне кнопку з написом «Подивитись», то перед ним з’являться результати останнього вводу – інтервали між сусідніми клавішами та час утримання кожної з них (рис. 3.2). Після того як користувач вводить пароль довільну кількість раз, необхідно натиснути відповідну кнопку «Зберегти» (рис. 3.3).

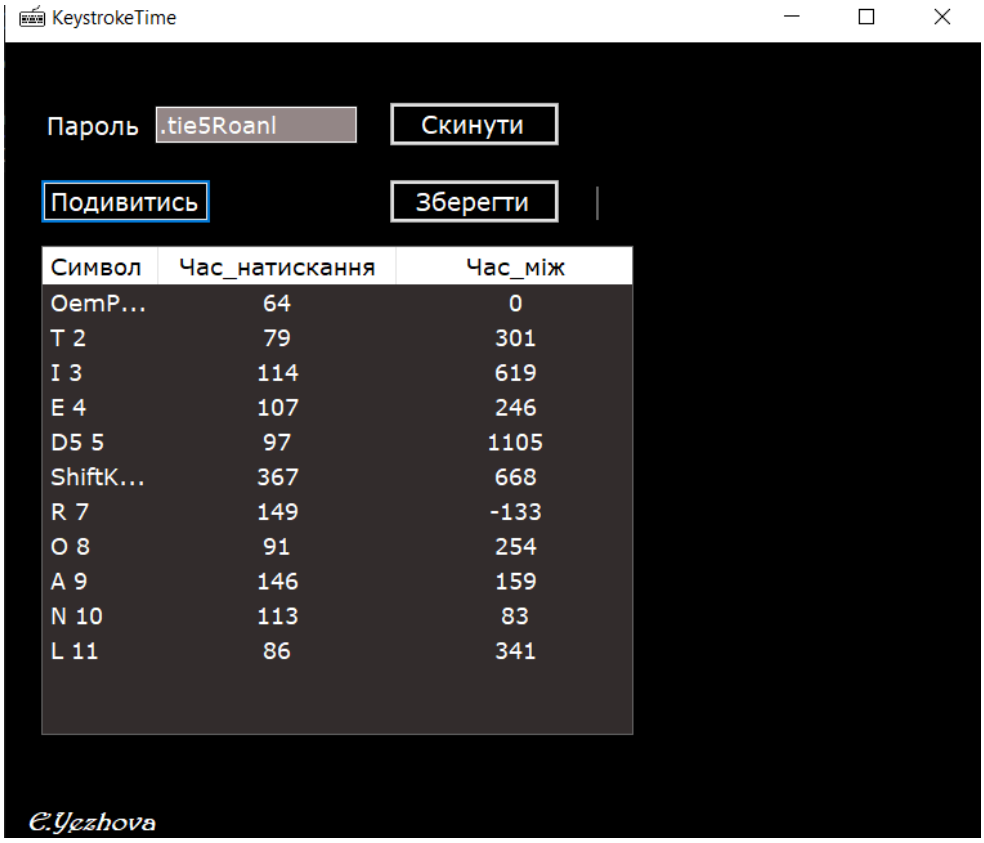


Рисунок 3.2 – Вхідні параметри вводу

Джерело: авторська розробка

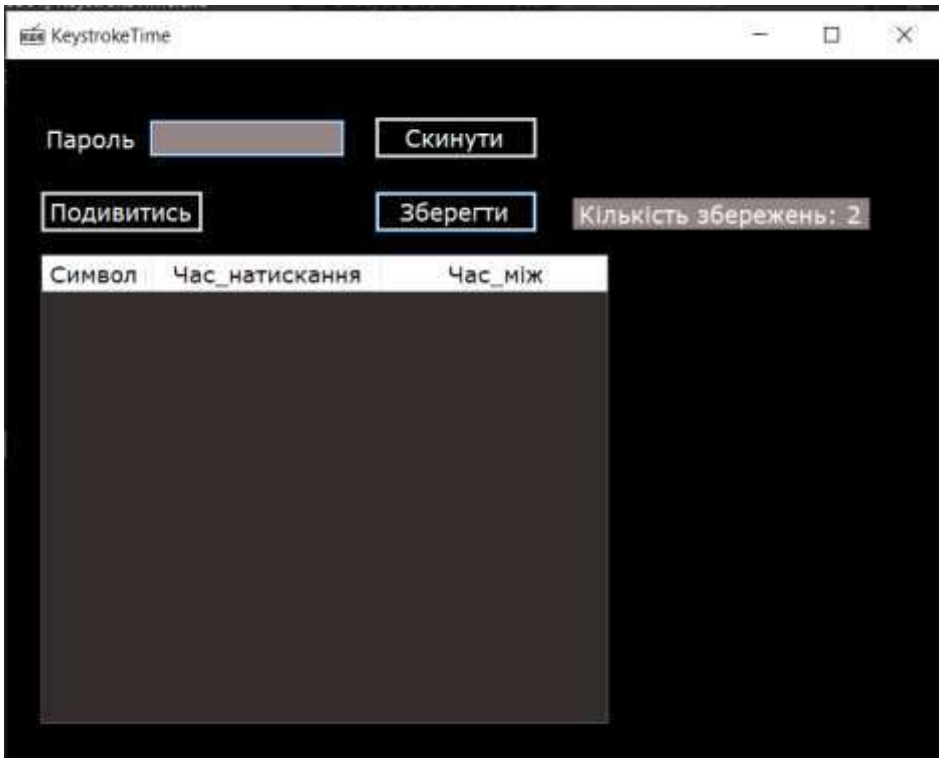


Рисунок 3.3 – Збереження зібраних даних

Джерело: авторська розробка

Після цього зібрані дані подано у вигляді таблиці з 34 стовпчиками. Кожен рядок даних відповідає інформації про час одноразового повторення пароля одним досліджуваним. Перший стовпчик, є унікальним ідентифікатором для кожного користувача (наприклад, s001 або s057) (рис. 3.4).

Другий стовпчик, sessionIndex, – це сеанс, під час якого введено пароль, у діапазоні від 1 до 8. Третій стовпчик rep, – це повторення пароля в сеансі, у діапазоні від 1 до 50.

Наступні 31 стовпчики містять інформацію про час введення пароля, а назва стовпчика вказує на вид даних про час. Наприклад, стовпець з назвою «H.period» (в загальному випадку H.key, де key – символ пароля) означає час утримання клавіші «.». Час утримання – це час від моменту натискання клавіші до моменту її відпускання.

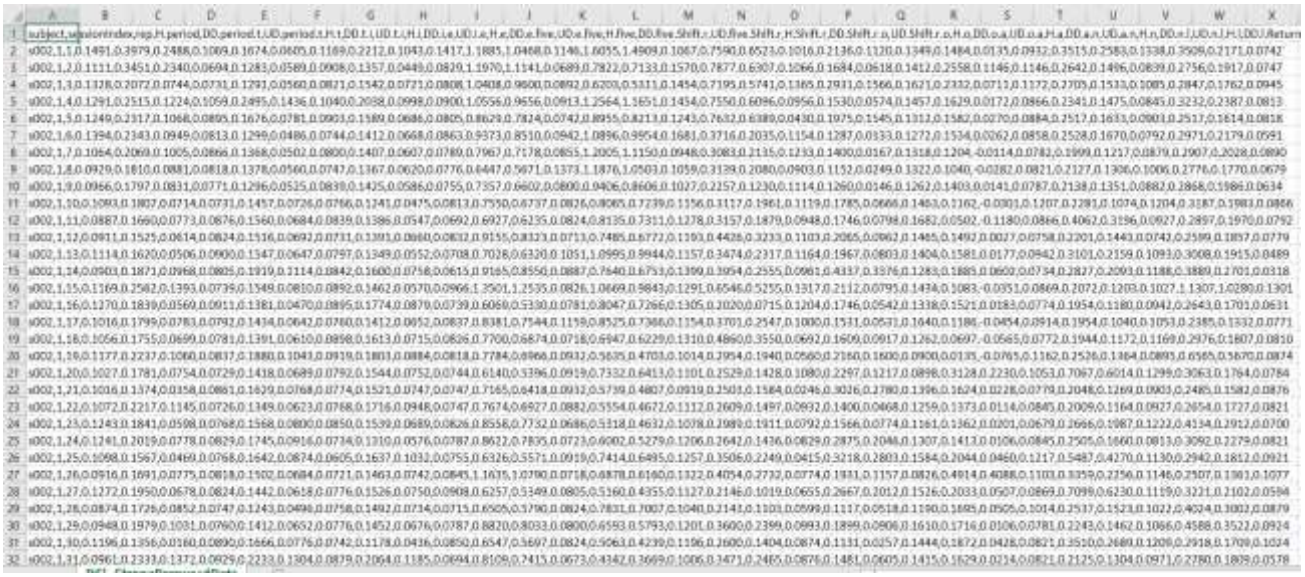


Рисунок 3.4 – Зображення вхідних даних

Джерело: за даними [42]

Іншим прикладом є стовпчик з назвою «DD.period.t» (в загальному випадку DD.key1.key2), що означає час від моменту натискання клавіші «.» до моменту натискання клавіші «t».

Стовпчик з назвою «UD.period.t» (в загальному випадку UD.key1.key2), він означає інтервал часу між натисканням клавіші «.» і натисканням клавіші «t», також відомий як час від моменту, коли відпущено «.», до моменту, коли натиснуто «t».

3.2 Етапи попередньої обробки для підготовки даних

Далі виконаємо попередній аналіз даних. Відобразимо розмір масиву даних (рис. 3.5), перші 5 рядків файлу (рис. 3.6), статистику даних (кількість спостережень, мінімальне, максимальне та середня значення, стандарте відхилення) (рис. 3.7) та загальну інформацію, включаючи індекс, тип, стовпці, ненульові значення та використання пам’яті (рис. 3.8).

```
df.shape
```

```
(20800, 34)
```

Рис. 3.5 – Розмір вхідних даних

Джерело: авторська розробка

```
df.head()
```

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	H.a	DD.
0	1	1	1	0.109	0.109	0.0	0.970	0.158	0.610	0.74	...	0.115	0.2
1	1	1	2	0.930	0.930	0.0	0.110	0.194	0.840	0.92	...	0.124	0.4
2	1	1	3	0.870	0.870	0.0	0.970	0.629	0.532	0.82	...	0.129	0.4
3	1	1	4	0.940	0.940	0.0	0.101	0.171	0.700	0.82	...	0.146	0.4
4	1	1	5	0.970	0.970	0.0	0.970	0.157	0.600	0.84	...	0.107	0.2

5 rows × 34 columns

Рисунок 3.6 – Відображення вхідних даних

Джерело: авторська розробка

```
df.describe()
```

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	
count	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	...	20800.000
mean	29.923077	4.500000	25.500000	0.100545	0.268030	0.167485	0.097119	0.170828	0.098184	0.094553	...	0.116
std	16.372093	2.291343	14.431217	0.078707	0.226237	0.225685	0.068033	0.124793	0.138238	0.100911	...	0.093
min	1.000000	1.000000	1.000000	0.001400	0.016700	-0.235000	0.008300	0.001100	-0.162100	0.003200	...	0.004
25%	16.750000	2.750000	13.000000	0.074700	0.145975	0.047200	0.066500	0.114200	0.028600	0.062300	...	0.082
50%	30.500000	4.500000	25.500000	0.090200	0.206200	0.105500	0.081600	0.141400	0.058800	0.077800	...	0.102
75%	43.250000	6.250000	38.000000	0.109600	0.309300	0.208600	0.101400	0.186000	0.098800	0.098700	...	0.123
max	57.000000	8.000000	50.000000	0.990000	12.506100	12.451700	0.990000	4.919700	4.799800	0.990000	...	2.035

8 rows × 34 columns

Рисунок 3.7 – Статистика даних

Джерело: авторська розробка

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 34 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   subject          20800 non-null  int64
 1   sessionIndex     20800 non-null  int64
 2   rep              20800 non-null  int64
 3   H.period         20800 non-null  float64
 4   DD.period.t      20800 non-null  float64
 5   UD.period.t      20800 non-null  float64
 6   H.t              20800 non-null  float64
 7   DD.t.i           20800 non-null  float64
 8   UD.t.i           20800 non-null  float64
 9   H.i              20800 non-null  float64
 10  DD.i.e           20800 non-null  float64
 11  UD.i.e           20800 non-null  float64
 12  H.e              20800 non-null  float64
 13  DD.e.five        20800 non-null  float64
 14  UD.e.five        20800 non-null  float64
 15  H.five           20800 non-null  float64
 16  DD.five.Shift.r  20800 non-null  float64
 17  UD.five.Shift.r  20800 non-null  float64
 18  H.Shift.r        20800 non-null  float64
 19  DD.Shift.r.o     20800 non-null  float64
 20  UD.Shift.r.o     20800 non-null  float64
 21  H.o              20800 non-null  float64
 22  DD.o.a           20800 non-null  float64
 23  UD.o.a           20800 non-null  float64
 24  H.a              20800 non-null  float64
 25  DD.a.n           20800 non-null  float64
 26  UD.a.n           20800 non-null  float64
 27  H.n              20800 non-null  float64
 28  DD.n.l           20800 non-null  float64
 29  UD.n.l           20800 non-null  float64
 30  H.l              20800 non-null  float64
 31  DD.l.Return      20800 non-null  float64
 32  UD.l.Return      20800 non-null  float64
 33  H.Return         20800 non-null  float64
dtypes: float64(31), int64(3)
memory usage: 5.4 MB
```

Рис. 3.8 – Загальна інформація

Джерело: авторська розробка

Для кращого відображення результатів, також необхідно розподілити дані за часом утримання – H.key (рис. 3.9), між натискання наступних однією за одною клавішами – DD.key1.key2 (рис. 3.10) та часом між відпусканням першої та наступної клавіши – UD.key1.key2 (рис. 3.11).

```
df[[x for x in df.columns if 'H' in x]].describe()
```

	H.period	H.t	H.i	H.e	H.five	H.Shift.r	H.o	H.a	H.n	H.l	H.Return
count	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000
mean	0.100545	0.097119	0.094553	0.092682	0.089753	0.098648	0.093861	0.116347	0.096850	0.108311	0.099491
std	0.078707	0.098033	0.100911	0.058813	0.163488	0.046682	0.066232	0.093349	0.077825	0.099656	0.091987
min	0.001400	0.009300	0.003200	0.002100	0.001400	0.001400	0.006900	0.004000	0.003700	0.003700	0.002900
25%	0.074700	0.066500	0.062300	0.060900	0.061200	0.070700	0.072000	0.062800	0.067800	0.077800	0.070200
50%	0.090200	0.081600	0.077800	0.083900	0.074700	0.094350	0.086900	0.102600	0.086050	0.094200	0.086300
75%	0.109000	0.101400	0.098700	0.104200	0.091800	0.118100	0.103200	0.123725	0.100300	0.112600	0.105100
max	0.990000	0.999000	0.990000	0.990900	0.990000	0.983000	0.994000	2.035300	0.990000	0.985000	0.970000

Рисунок 3.9 – Розбиття вхідних даних за часом утримання

Джерело: авторська розробка

```
df[[x for x in df.columns if 'DD' in x]].describe()
```

	DD.period.t	DD.t.i	DD.i.e	DD.e.five	DD.five.Shift.r	DD.Shift.r.o	DD.o.a	DD.a.n	DD.n.l	DD.l.Return
count	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000
mean	0.268039	0.170820	0.160956	0.375497	0.436610	0.252449	0.159856	0.153098	0.204239	0.321287
std	0.226237	0.124793	0.225673	0.264322	0.260711	0.177171	0.110625	0.111188	0.152119	0.224340
min	0.018700	0.001100	0.001400	0.001300	-0.900000	0.049400	0.001200	0.001100	0.001300	0.008300
25%	0.145875	0.114200	0.009675	0.215400	0.305800	0.156000	0.106800	0.096600	0.120300	0.210400
50%	0.206200	0.141400	0.122300	0.287300	0.376200	0.201300	0.132400	0.128400	0.172800	0.262300
75%	0.309300	0.186000	0.175700	0.454225	0.485300	0.285100	0.170700	0.177000	0.229800	0.349300
max	12.506100	4.919700	25.987300	4.961800	8.370200	4.152300	2.856700	3.327800	4.025200	5.883600

Рисунок 3.10 – Розбиття вхідних даних за часом між натискання наступних однією за одною клавішами

Джерело: авторська розробка

```
df[[x for x in df.columns if 'UD' in x]].describe()
```

	UD.period.t	UD.t.i	UD.i.e	UD.e.five	UD.five.Shift.r	UD.Shift.r.o	UD.o.a	UD.a.n	UD.n.l	UD.l.Return
count	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000	20800.000000
mean	0.167485	0.090184	0.065436	0.290499	0.358445	0.143609	0.072737	0.052458	0.117086	0.225734
std	0.225865	0.139238	0.237187	0.267548	0.264864	0.204137	0.116814	0.125911	0.164831	0.228961
min	-0.235800	-0.162100	-0.160000	-0.150500	-0.990000	-0.900000	-0.228700	-0.800000	-0.700000	-0.124500
25%	0.047200	0.020000	0.000000	0.133375	0.227000	0.052500	0.017500	-0.006200	0.025800	0.114700
50%	0.105500	0.058900	0.042500	0.201050	0.306000	0.100000	0.045500	0.023000	0.066000	0.180900
75%	0.209600	0.099500	0.098100	0.373750	0.408075	0.188000	0.083800	0.073400	0.140300	0.253300
max	12.451700	4.792900	25.915000	4.852700	8.296000	4.012000	2.815200	2.524200	3.978200	5.836400

Рисунок 3.11 – Розбиття вхідних даних за часом між відпусканням першої та наступної клавіши

Джерело: авторська розробка

Отже, як видно з рисунків 3.9–3.11, значення UD.key1.key2 може бути від’ємним, що дає змогу дійти висновку, що користувач натискає наступну клавішу, ще до того як відпустив попередню, а також висунути припущення, що H.key плюс UD.key1.key2 дорівнює DD.key1.key2 (рис. 3.12).

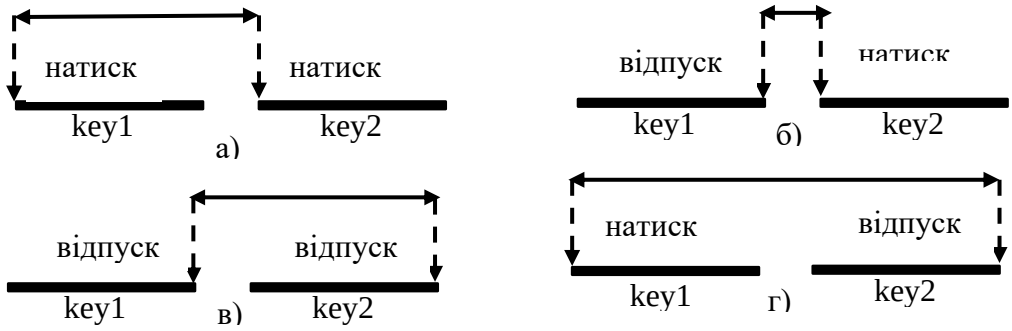


Рисунок 3.12 – Особливості динаміки натискання клавіш: а) час між натисканням двох кнопок пароля, б) час між відпусканням попередньої та натисканням наступної кнопки, в) час між відпусканням попередньої та наступної кнопки, г) загальний час

Джерело: авторська розробка

Перший експеримент полягає в тому, щоб побачити, чи є якась різниця між цими 3 основними групами. На рисунку 3.13 – 3.14 кожна лінія – це рядок, а кожен користувач містить 400 таких рядків.

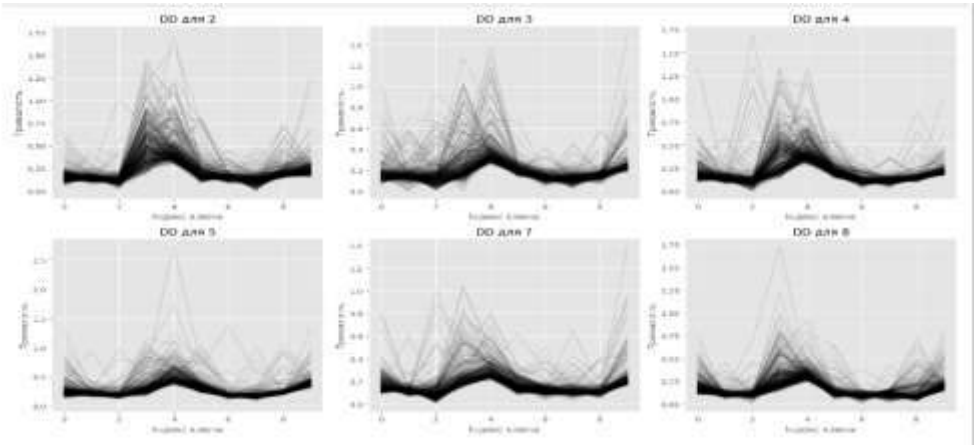


Рисунок 3.13 – Порівняння DD.key1.key2 для різних користувачів

Джерело: за даними [41]

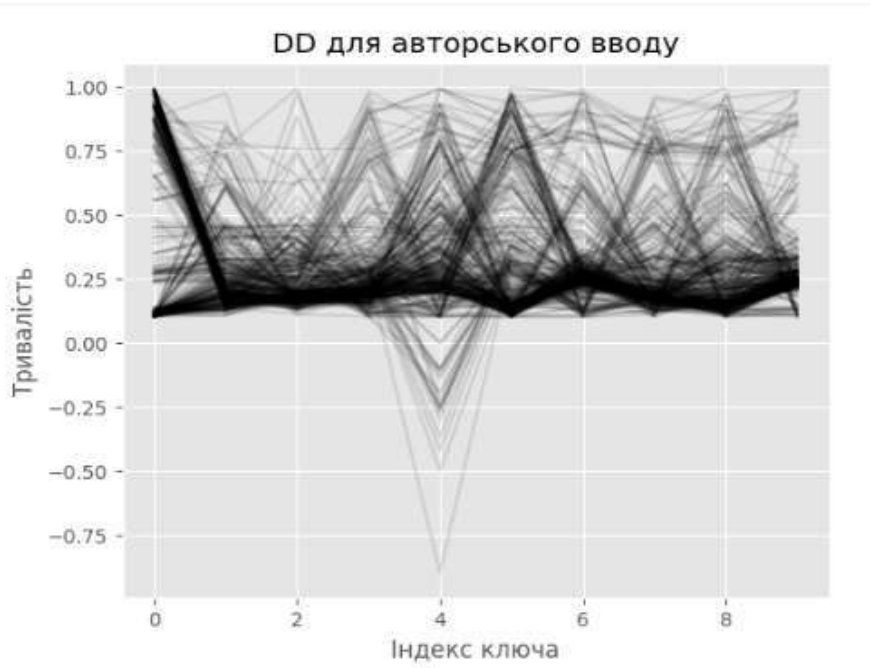


Рисунок 3.14 – Графік DD.key1.key2 для авторського вводу

Джерело: авторська розробка

З рисунку видно, що більшість записів мають схожий шаблон у межах одного користувача, що може свідчити про те, що один і той самий учасник має тенденцію до однакового набору тексту.

Це спостереження добре підходить для експерименту з класифікації. Однак, якщо порівнювати на одному рисунку шість користувачів та авторський ввід, результати показують, що всі шість також мають схожий шаблон, а ось авторський ввід відрізняється, це може бути пов'язано з тим, що всі користувачі вводять однакові символи, але автор вводив даний пароль перший раз.

Далі порівнюються середні значення натискання DD.key1.key2 серед тих самих користувачів. На рисунку 3.15 можна помітити невелику різницю, що показує, що за ознакою натискання клавіші (DD.key1.key2) можна розрізняти користувачів, що може бути використано в подальшому для класифікації.

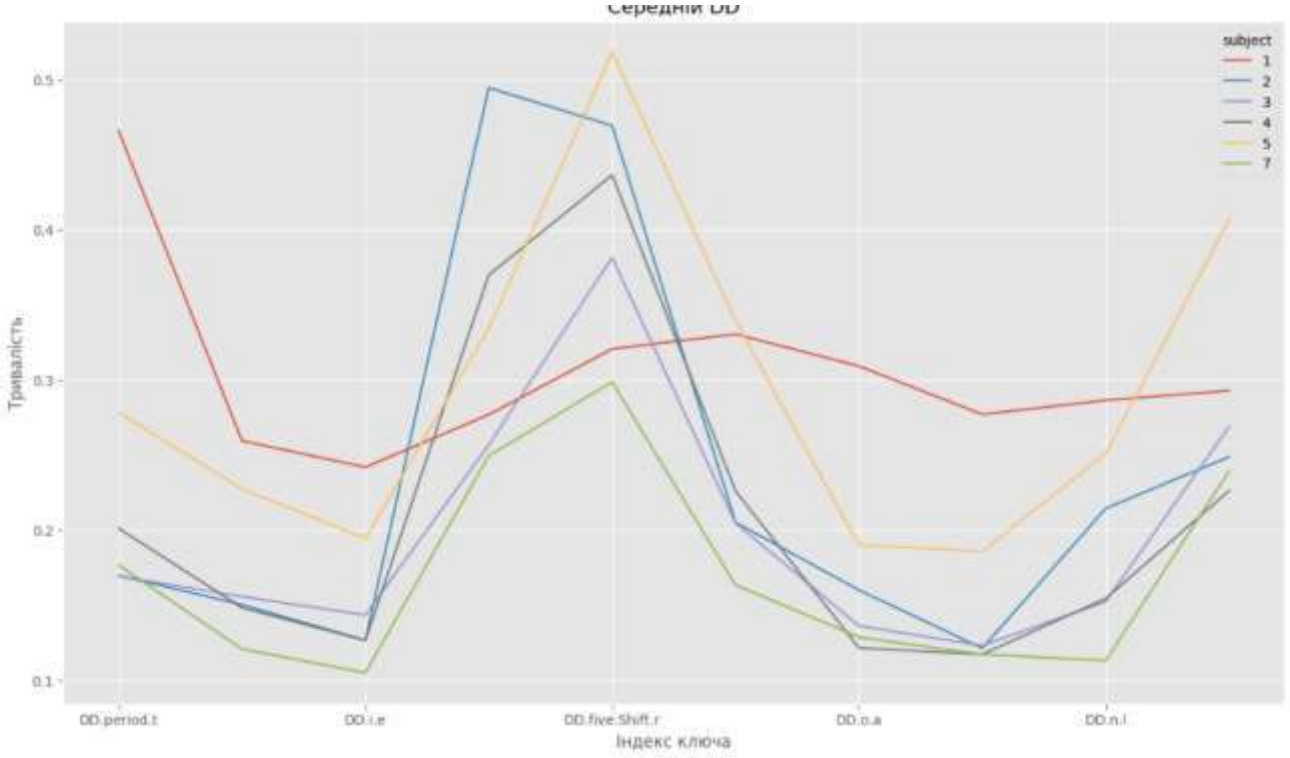


Рисунок 3.15 – Середнє значення DD.key1.key2 для різних користувачів

Джерело: авторська розробка

Далі виконано такі самі дослідження, але для часу H.key (рис. 3.16-3.18) та UD.key1.key2 (рис. 3.19-3.21).

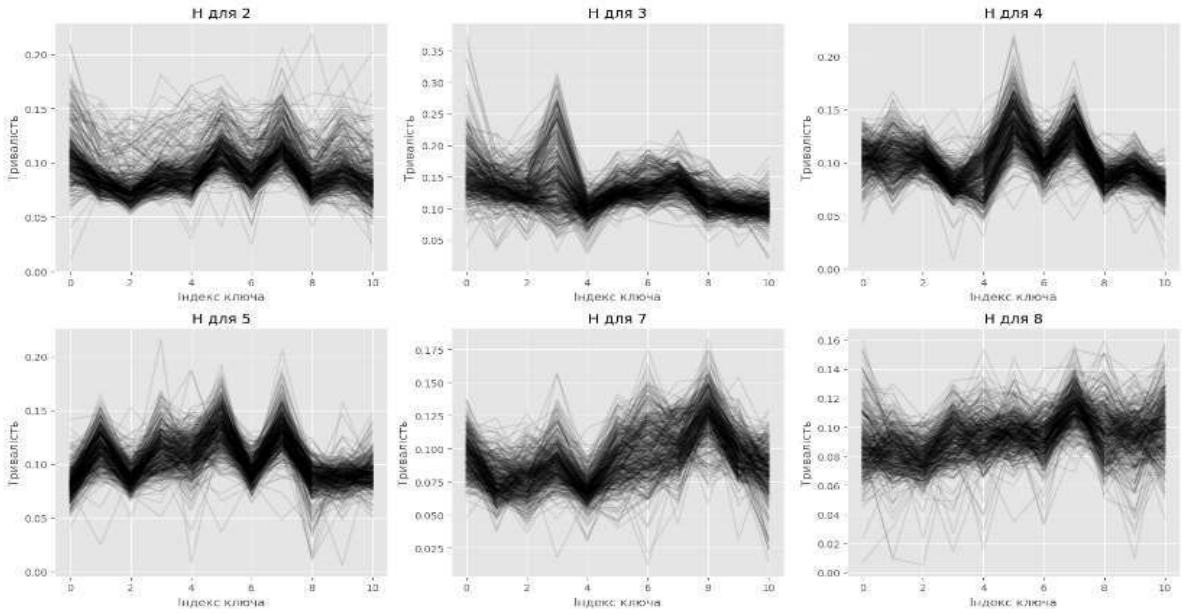


Рисунок 3.16 – Порівняння H.key для різних користувачів

Джерело: за даними [42]

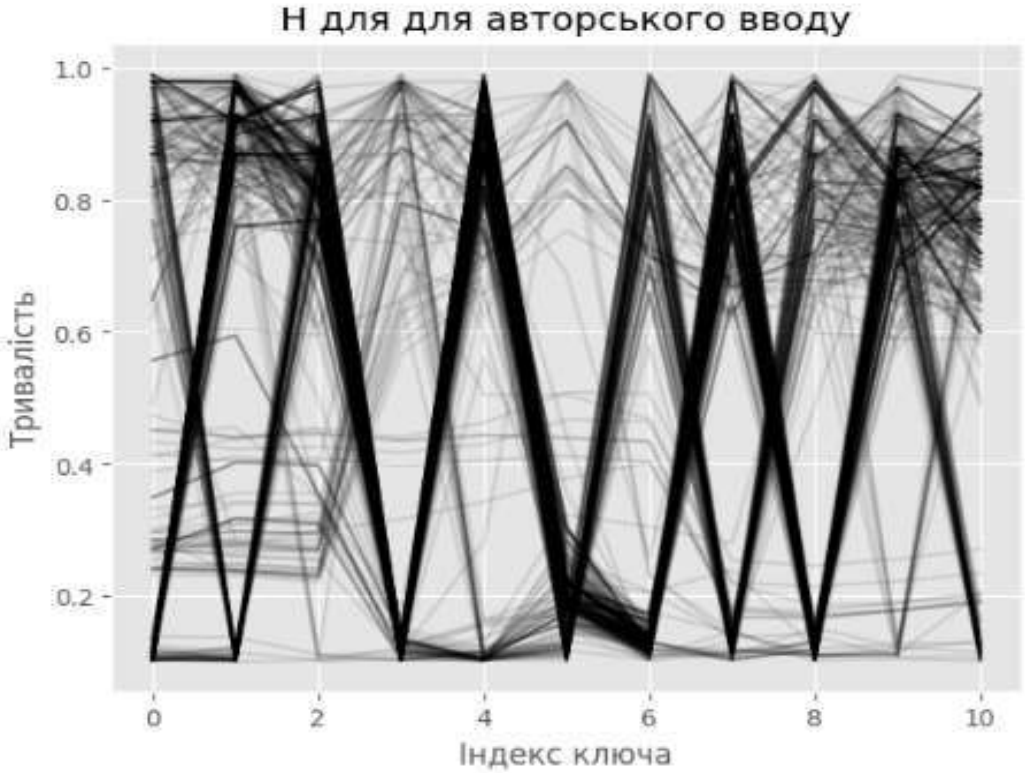


Рисунок 3.17 – Графік H.key для авторського вводу

Джерело: авторська розробка

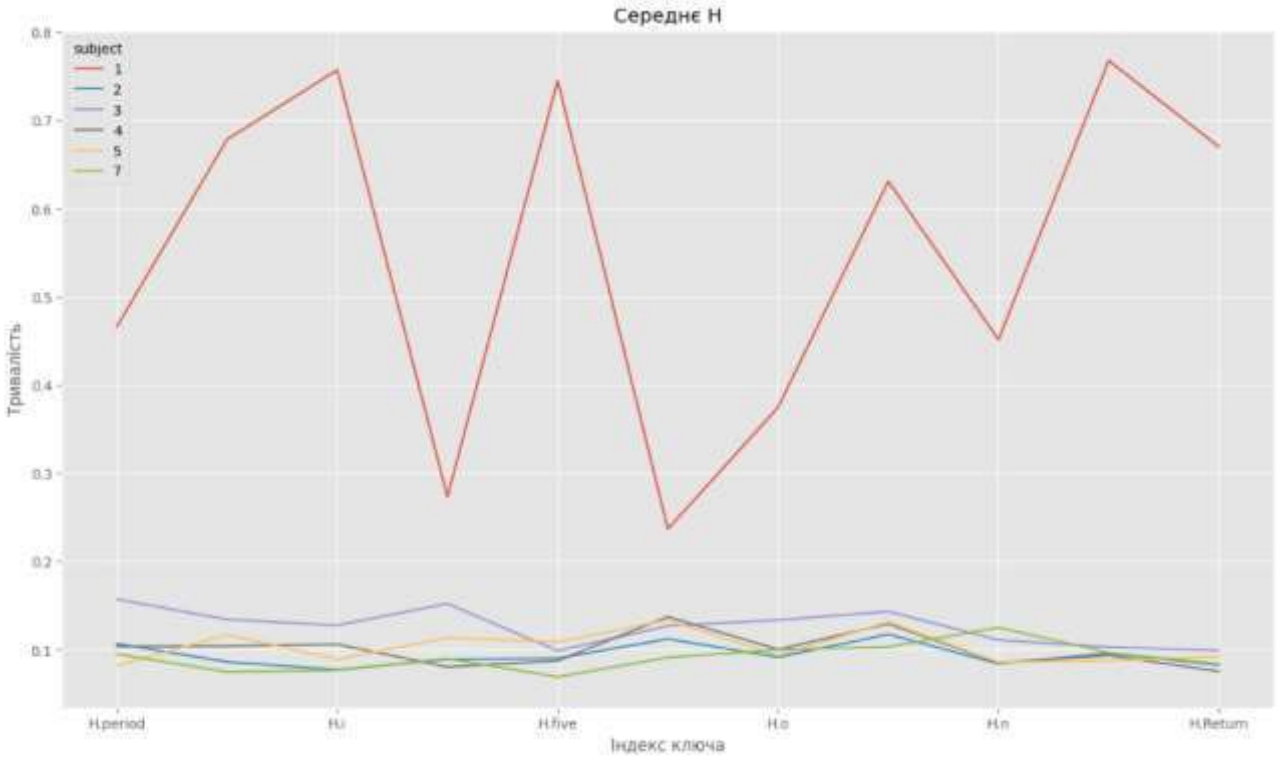


Рисунок 3.18 – Середнє значення H.key для різних користувачів

Джерело: авторська розробка

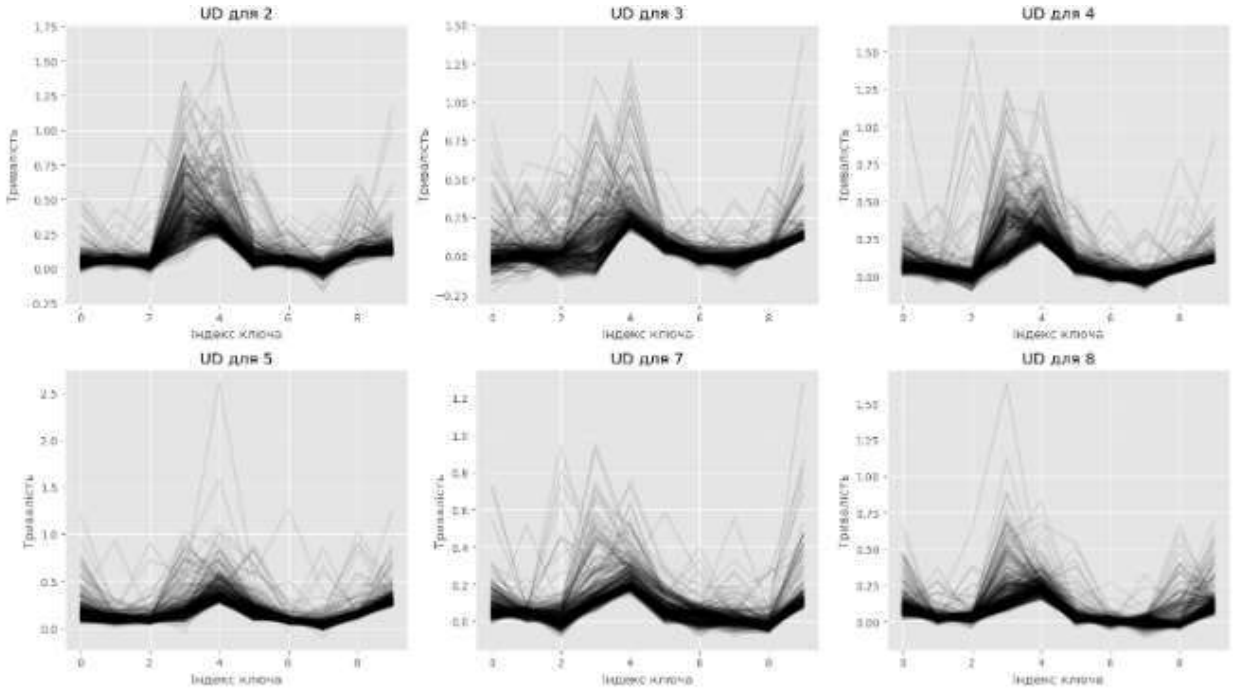


Рисунок 3.19 – Порівняння UD.key1.key2 для різних користувачів

Джерело: за даними [42]

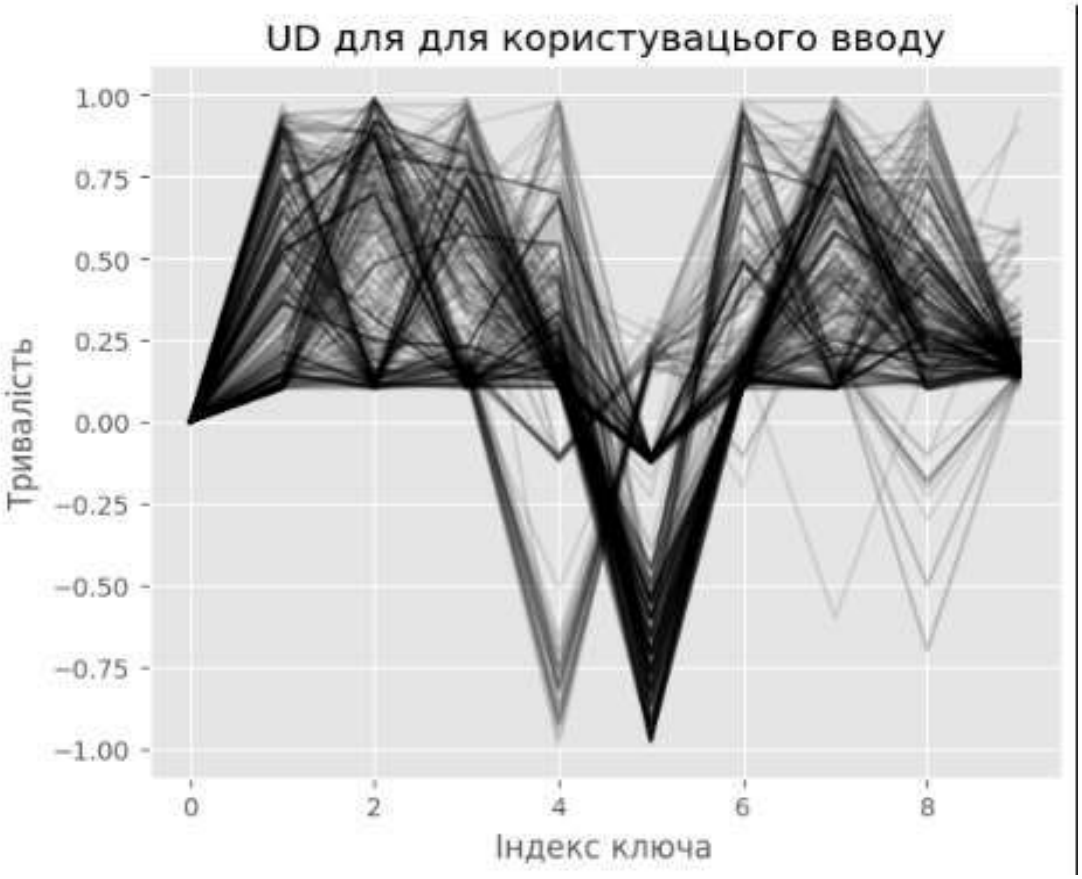


Рисунок 3.20 – Графік UD.key1.key2 для авторського вводу

Джерело: авторська розробка

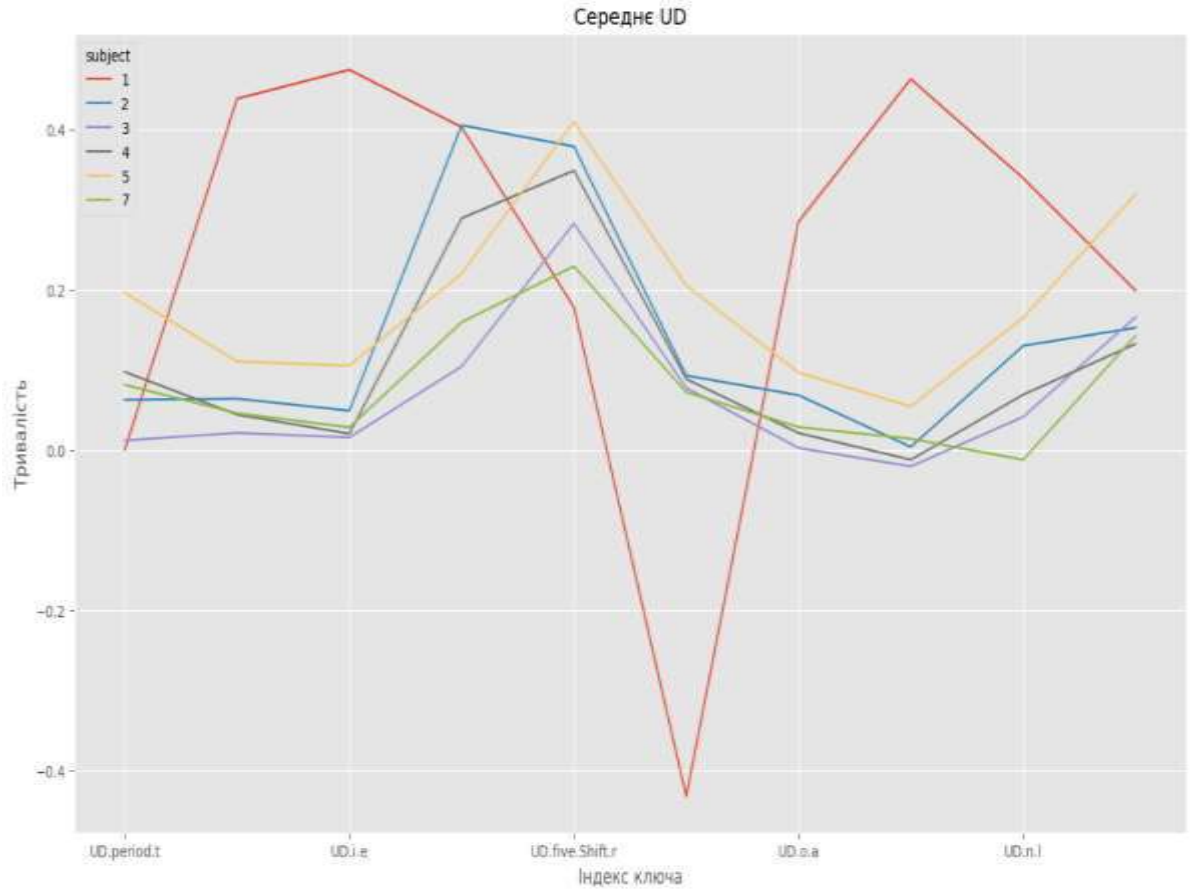


Рисунок 3.21 – Середнє значення UD.key1.key2 для різних користувачів

Джерело: авторська розробка

Якщо порівнювати результати часу DD.key1.key2 (рис. 3.16-3.18) та H.key (рис. 3.19-3.21), можна дійти висновку, що тривалість натискання (H.key) може бути важливою характеристикою, оскільки за нею набагато простіше класифікувати всіх користувачів порівняно з іншими параметрами.

Для з’ясування залежності параметрів побудовано теплову карту кореляції (рис. 3.22). Теплова карта є графічним відображенням числових даних, у якому колір використовується для позначення значень.

Дослідивши отриману карту, можна дійти висновку, що існує сильна залежність між UD та DD параметрами на кожному символі, що підтверджує, що H.key плюс UD.key1.key2 дорівнює DD.key1.key2.

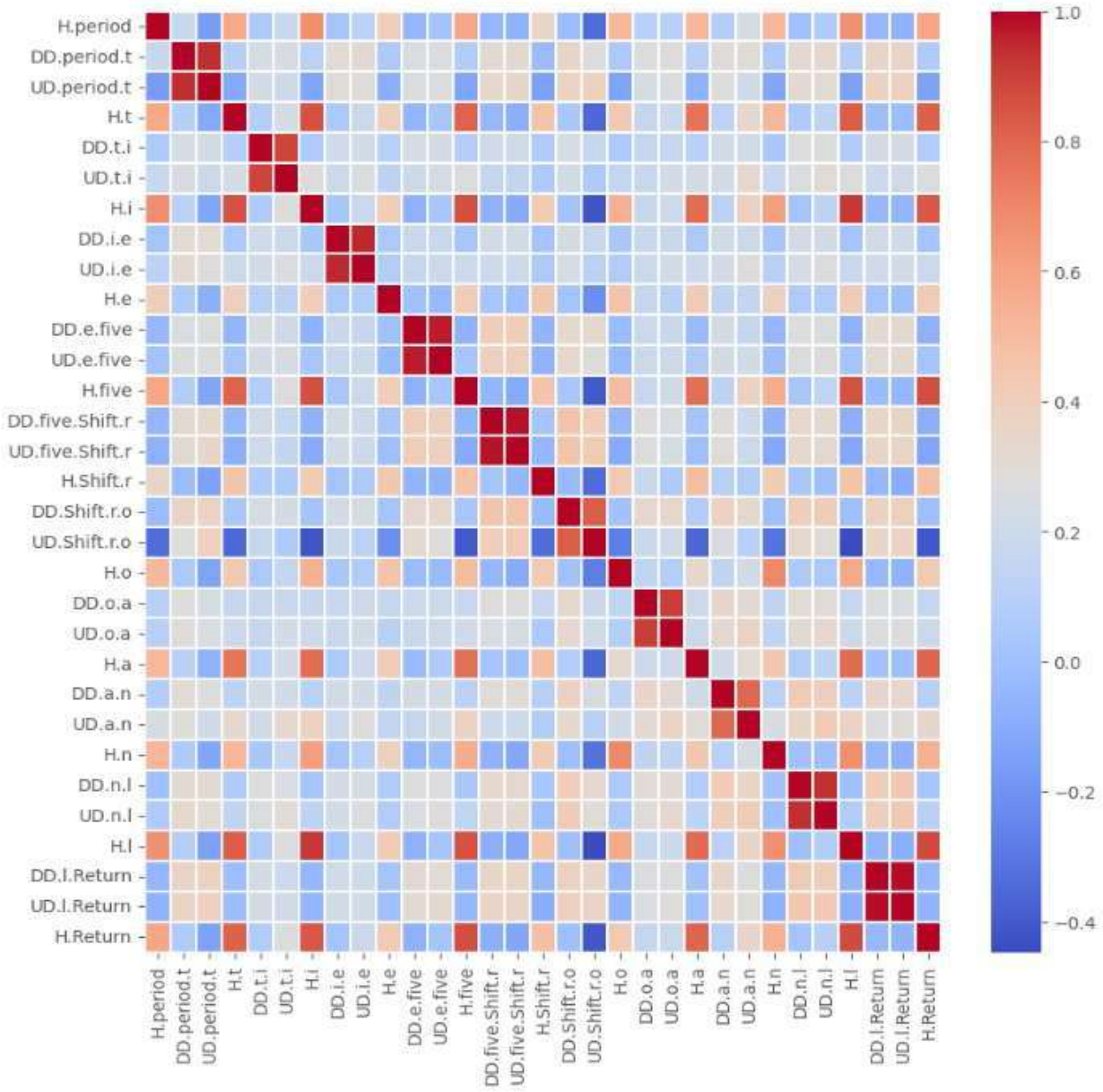


Рисунок 3.22 – Теплова карта кореляції параметрів KD

Джерело: авторська розробка

3.3 Висновок за розділом

У цьому розділі проведено аналіз набору даних для аутентифікації користувачів на основі динаміки натискання клавіш.

Спершу визначено розмір набору даних, що дозволило отримати уявлення про обсяг та кількість даних, які потрібно обробити. Подальший аналіз статистики надав інформацію про розподіл даних, включаючи мінімальні та максимальні значення, середні та стандартні відхилення. Ця

інформація була корисною для розуміння різноманітності та характеристик даних.

Також використано розбиття даних за різними часовими параметрами, включаючи час натискання клавіш (H.key), час між натисканням послідовних клавіш (DD.key1.key2) та час між відпусканням однієї клавіші та натисканням іншої (UD.key1.key2). Ця деталізація дозволила виявити подібність між користувачами та виділити ключові параметри, які можуть бути використані для подальшого аналізу та розробки моделі аутентифікації.

РОЗДІЛ 4
РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ З НАВЧАННЯ ТА ТЕСТУВАННЯ

4.1 Процес навчання та налаштування параметрів

В якості навчальних даних (training dataset), тобто даних, які безпосередньо використовуються для навчання мережі, відібрано 70% з отриманого раніш датасету та з них відокремлено 10%, що будуть використані як валідаційні дані (validation dataset), тобто дані для оцінки точності мережі. Для забезпечення незміщеної оцінки остаточної відповідності мережі на навчальному наборі даних, використано 30 % тестових даних (test dataset). За допомогою бібліотеки Keras Tuner та безпосередньо методу RandomSearch виконано пошук найбільш раціональних гіперпараметрів з точки зору максимальної точності. Як дані, що використовувались для підбору гіперпараметрів використано training dataset.

При створенні моделі для аутентифікації за KD, підібрано такі гіперпараметри:

1) conv_1_filter – кількість фільтрів для першого шару 1DConv (від 16 до 128, з кроком 16);

2) conv_1_kernel – довжина згортки для першого шару 1DConv (2 або 3);

3) conv_1_activation – FA для першого шару 1DConv (ReLU – R, Tanh – T, або sigmoid – S);

4) conv_2_filter – кількість фільтрів для другого шару 1DConv (від 16 до 128, з кроком 16);

5) conv_2_kernel – довжина згортки для другого шару 1DConv (2 або 3);

6) conv_2_activation – FA для другого шару 1DConv (ReLU – R, Tanh – T, або sigmoid – S);

7) LSTM_units – кількість нейронів в LSTM (від 16 до 128, з кроком 16);

8) dense_1_units – кількість нейронів в Dense (від 16 до 128, з кроком 16);

9) dense_activation – FA для другого шару Dense (ReLU – R або Tanh – T);

10) dropout_rate – відсоток ймовірності відсівання (від 0,1 до 0.9, з кроком 0,1).

Загалом виконано 50 випробувань по 100 епох у кожному, значення точності збільшувалось під час цього процесу з кожною епохою.

Десять найкращих випробувань обрано на основі набору налаштувань гіперпараметрів, що дали найкращі значення точності зазначено в таблиці 4.1.

Таблиця 4.1 – Раціональні гіперпараметри для максимальної точності

Гіперпараметр	Випробування									
	1	2	3	4	5	6	7	8	9	10
conv_1_filter	112	64	32	48	128	16	80	32	80	80
conv_1_kernel	2	2	2	3	2	2	2	2	2	3
conv_1_activation	R	R	R	S	R	S	R	T	R	T
conv_2_filter	64	112	32	32	48	64	64	96	112	128
conv_2_kernel	2	2	2	3	2	3	2	2	2	3
conv_2_activation	R	R	T	R	S	R	R	R	R	R
LSTM_units	112	32	112	96	128	112	96	80	64	32
dense_1_units	48	80	112	48	112	128	64	80	80	80
dense_activation	T	T	T	T	T	R	R	T	R	T
dropout_rate	0.1	0.1	0.3	0.1	0.3	0.1	0.1	0.4	0.3	0.3
Точність	0.9988	0.9969	0.9968	0.9955	0.9954	0.9937	0.9901	0.9898	0.9895	0.9893

Як видно з таблиці 4.1 перші 7 найкращих випробувань мають майже однакову точність на рівні 0,99. Тому обрано гіперпараметри (рис. 4.1) зважаючи не тільки на значення точності, а й на раціональне співвідношення її з не перевантаженням мережі.

```
Trial 09 summary
Hyperparameters:
conv_1_filter: 64
conv_1_kernel: 2
conv_1_activation: relu
conv_2_filter: 112
conv_2_kernel: 2
conv_2_activation: relu
LSTM_units: 32
dense_1_units: 80
dense_activation: tanh
dropout_rate: 0.1
Score: 0.9968711733818054
```

Рисунок 4.1 – Раціональні гіперпараметри для максимальної точності

Джерело: авторська розробка

Далі перевірено чи вливає розмір обраного розміру пакета (batch size) на точність мережі. Розмір пакета визначає кількість зразків, які використано в одній епосі для навчання мережі.

На рисунку 4.2 наведено графік залежності точності від розміру пакета.

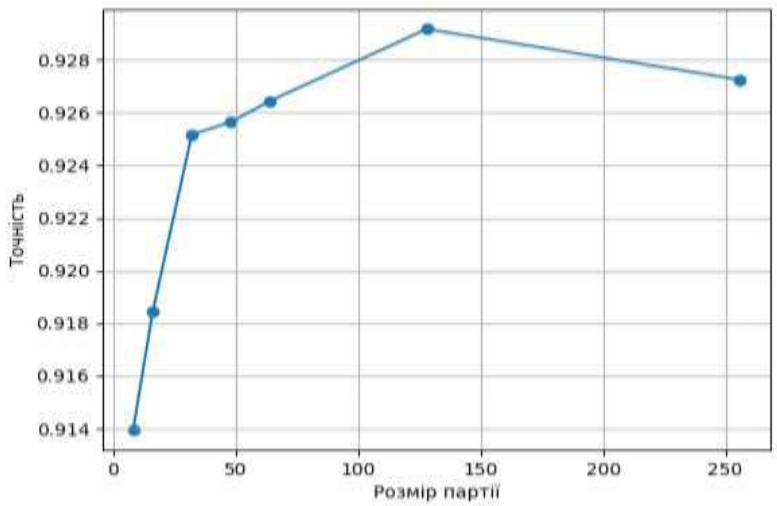


Рисунок 4.2 – Залежність точності від розміру пакета

Джерело: авторська розробка

Як видно з рисунку найбільша точність досягається при розмірі пакета 128. Також є необхідність визначення оптимальної кількості епох (рис. 4.3 – 4.4).

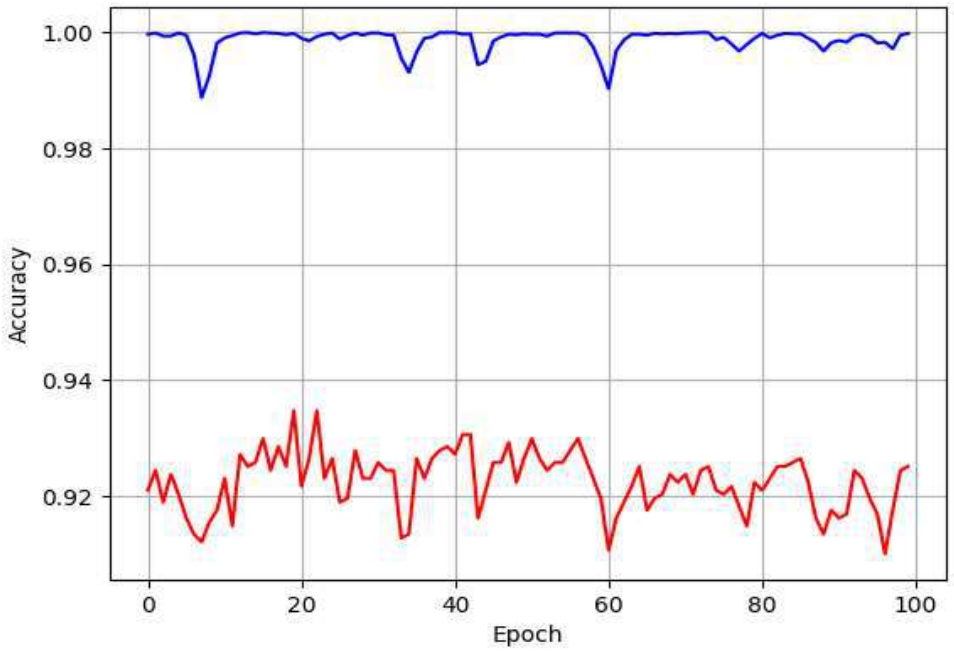


Рисунок 4.3 – Залежність точності від кількості епох

Джерело: авторська розробка

З рисунків 4.3 видно, що значення точності не є постійнозростаючим при збільшенні кількості епох та значення втрат (рис. 4.4) не завжди спадає, тобто необхідно визначити раціональну кількість епох при яких не починається перенавчання моделі. Саме тому при навчанні мережі доцільно використовувати ранню зупинку, яку детально описано в 1 розділі роботи. Основний параметром ранньої зупинку встановлено відсутність збільшення точності при 5 епохах. Залежність точності та втрат від кількості епох наведено на рисунках 4.5 – 4.6.

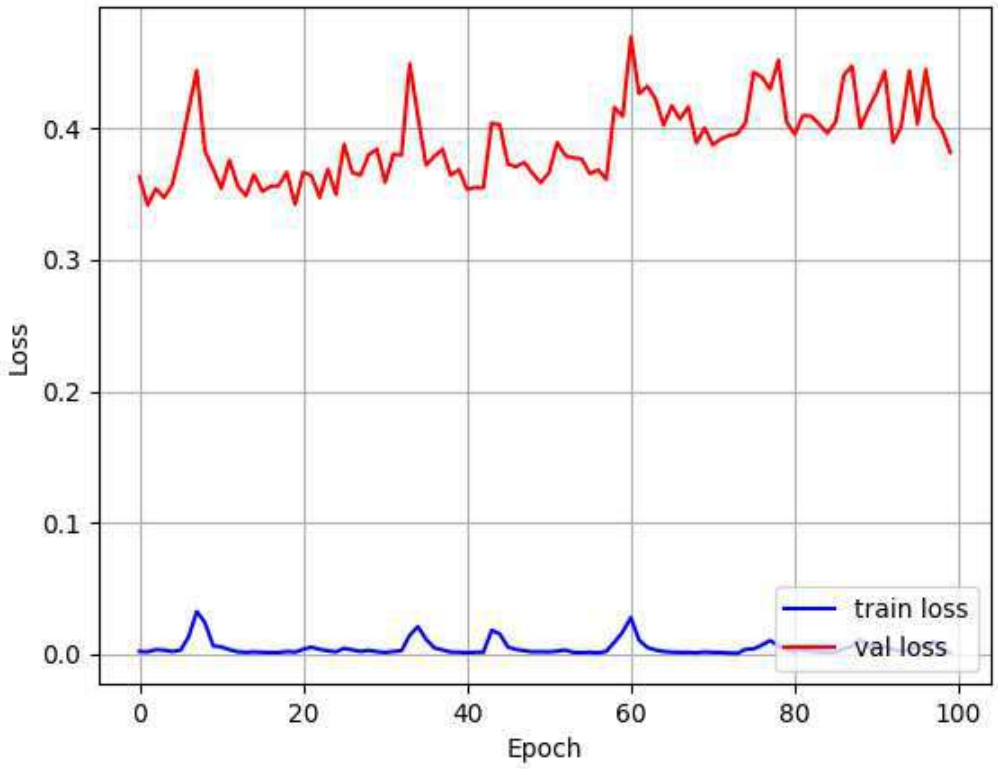


Рисунок 4.4 – Залежність втрат від кількості епох

Джерело: авторська розробка

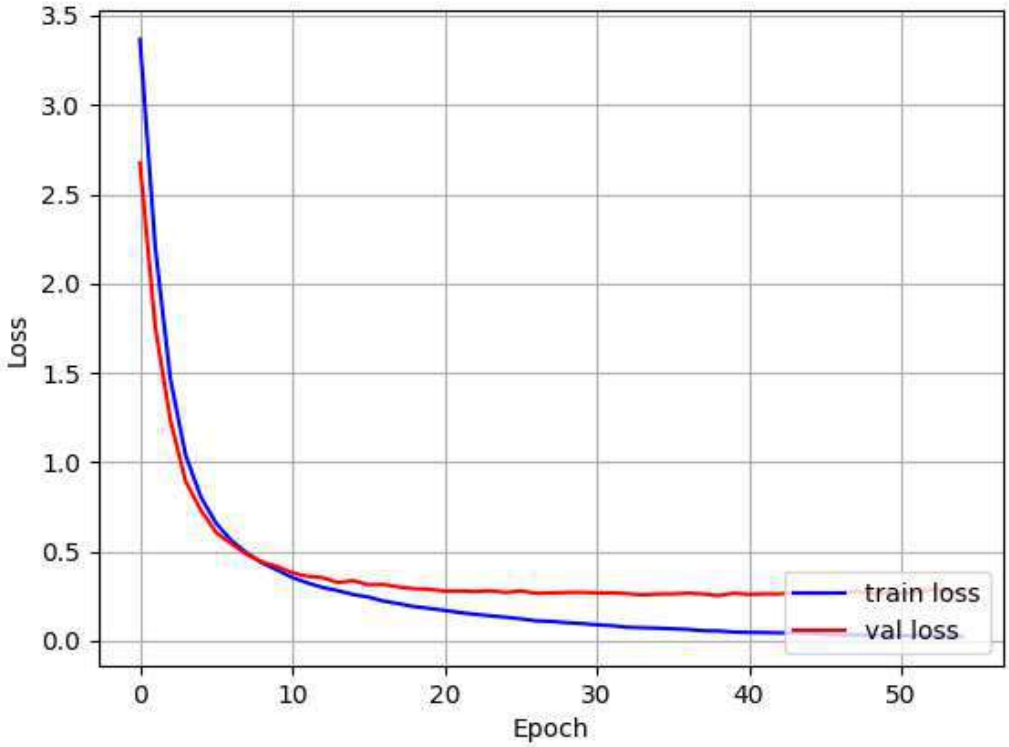


Рисунок 4.5 – Залежність втрат від кількості епох при ранній зупинці

Джерело: авторська розробка

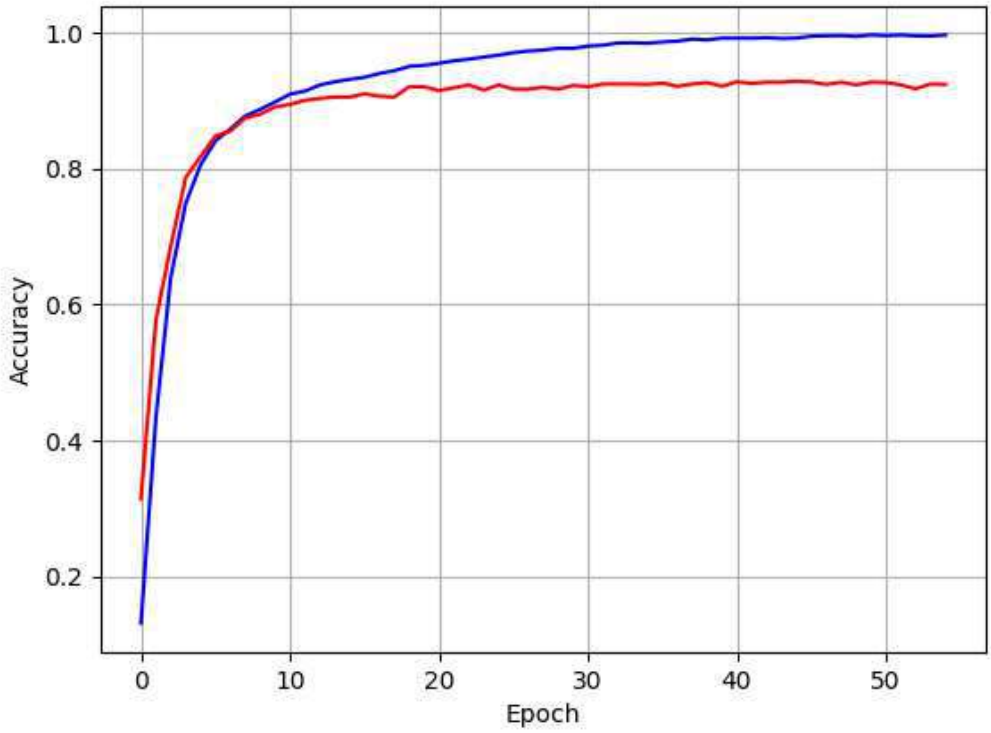


Рисунок 4.6 – Залежність точності від кількості епох при ранній зупинці

Джерело: авторська розробка

4.2 Тестування та оцінювання

Якість запропонованої моделі оцінено на основі метрик, що було описані раніш, таких як accuracy, precision, recall та F1-міра (рис. 4.7).

```
[0.2559188902378082, 0.9307692050933838]
Accuracy = 0.9307692307692308
Precision = 0.9314594819701478
Recall = 0.9307692307692308
F1 Score = 0.9305327311472332
```

Рисунок 4.7 – Розрахунок метрик моделі

Значення precision, recall та F1-міри для кожного користувача також отримано за допомогою classification_report (рис. 4.8).

	precision	recall	f1-score	support
0	1.00	0.99	1.00	117
1	0.84	0.85	0.85	119
2	0.91	0.96	0.93	113
3	0.93	0.92	0.92	125
4	0.97	0.97	0.97	116
5	0.91	0.83	0.87	139
6	0.94	0.84	0.88	141
7	0.98	0.97	0.97	132
8	0.93	0.91	0.92	128
9	0.92	0.93	0.92	145
…	…	…	…	…
44	0.89	0.91	0.90	121
45	0.90	0.90	0.90	122
46	0.98	0.98	0.98	122
47	0.96	0.98	0.97	122
48	0.90	0.97	0.93	119
49	0.97	0.98	0.97	117
50	0.98	0.90	0.94	123
51	0.94	0.81	0.87	111
accuracy			0.93	6240
macro avg	0.93	0.93	0.93	6240
weighted avg	0.93	0.93	0.93	6240

Рисунок 4.8 – Звіт основних метрик класифікації

Джерело: авторська розробка

На рисунку 4.9 наведено порівняння значень перерахованих вище метрик оцінювання.

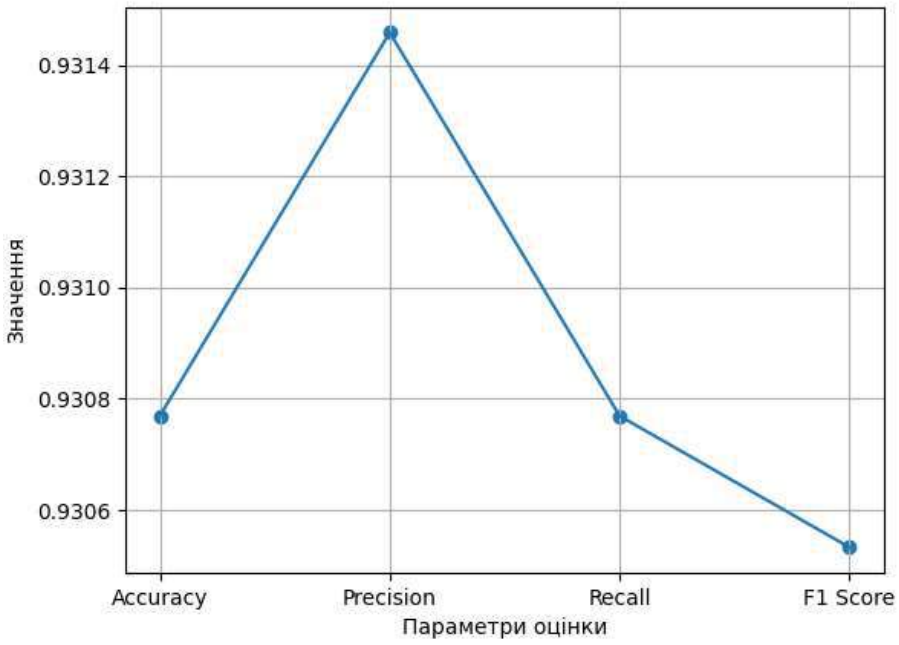


Рисунок 4.9 – Порівняння параметрів оцінювання

Джерело: авторська розробка

Також для оцінки точності мережі класифікації використано ROC-крива та AUC. Побудована крива наведена на рисунку 4.10.

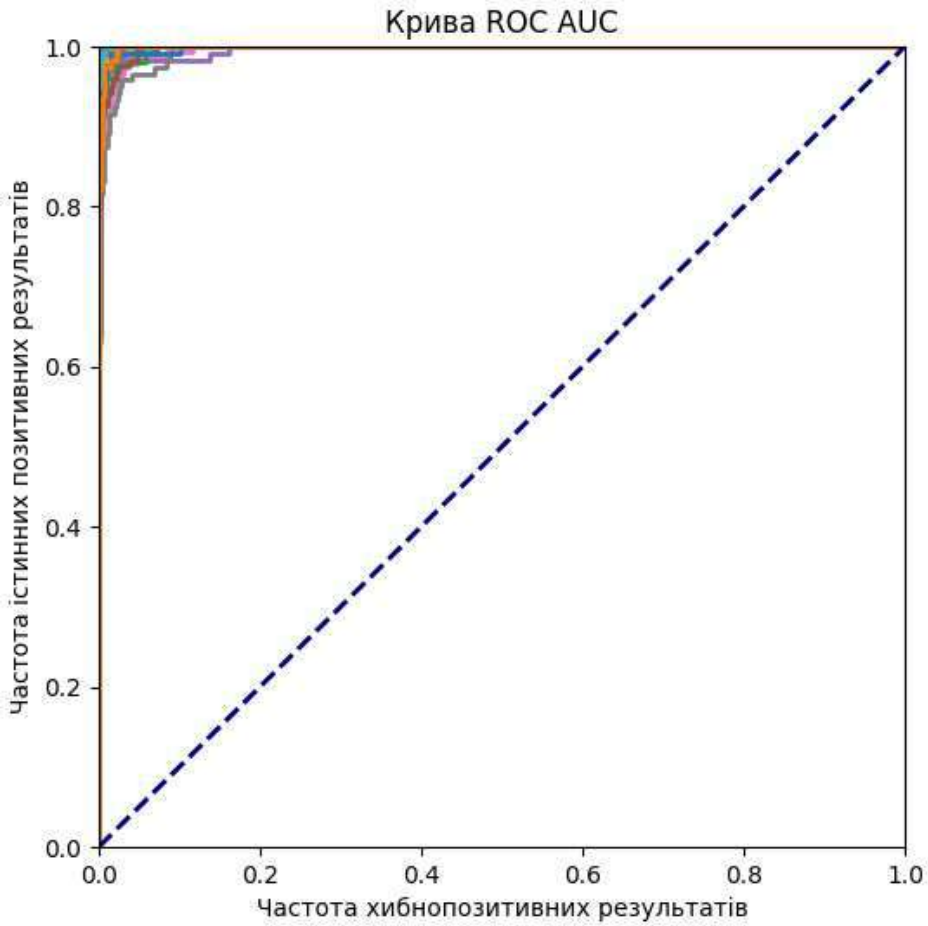


Рисунок 4.10 – Графік ROC-кривої

Джерело: авторська розробка

З рисунку видно, що ROC-крива охоплює верхній лівий кут, що дає підставу вважати оцінювану модель досить точною.

Значення AUC для оцінюваної моделі мережі наведено на рисунку 4.11.

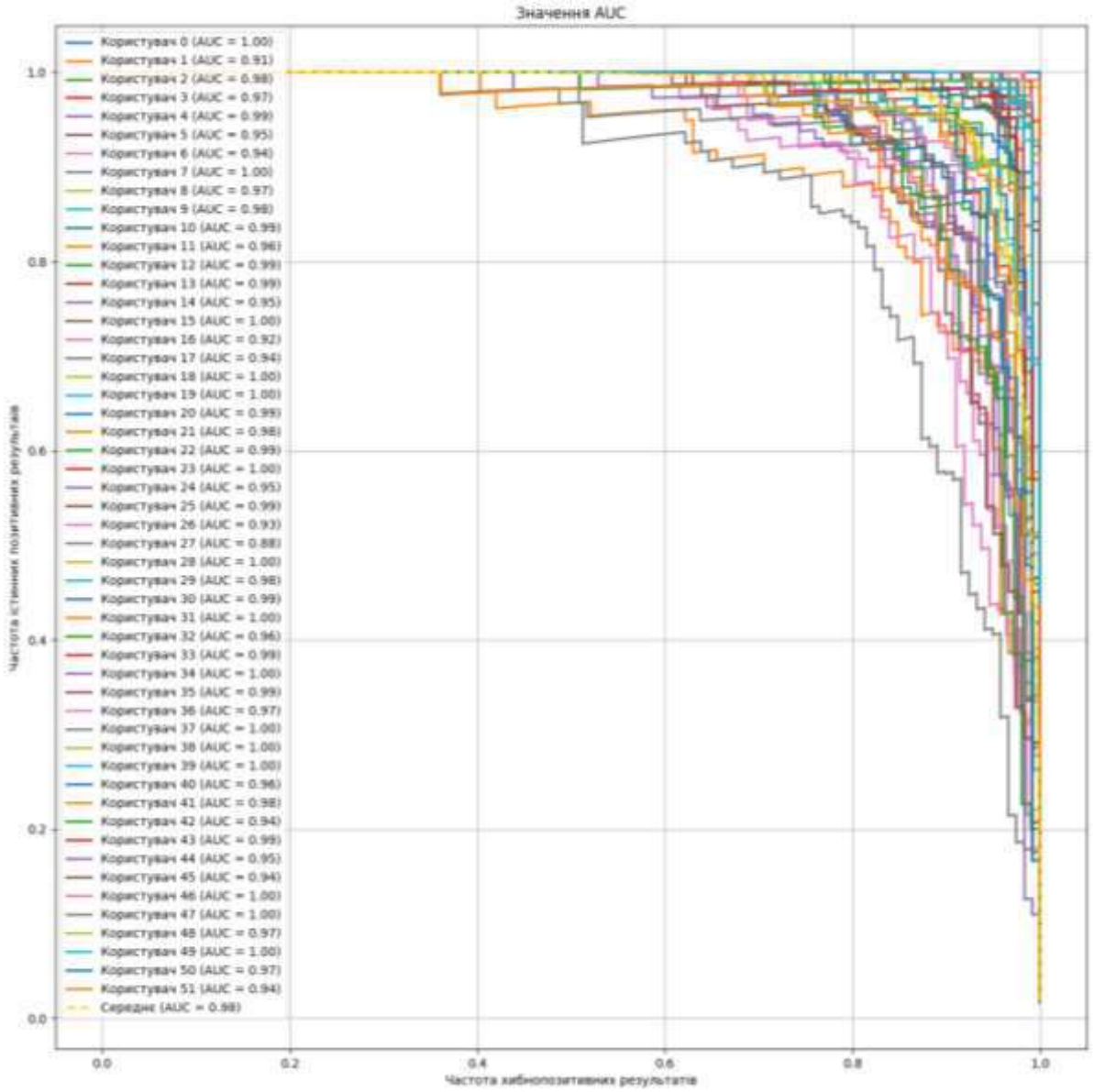


Рисунок 4.11 – Побудова AUC

Джерело: авторська розробка

Як видно з рисунку значення AUC знаходяться близько до 1, що дає підставами стверджувати, що модель досягає високої точності розрізнення між відповідним класом та усіма іншими класами.

Також для оцінювання використано confusion matrix з кількістю класів для класифікації 52.

Матриця плутанини для користувачів з 001 до 012 наведено на рисунку 4.12.

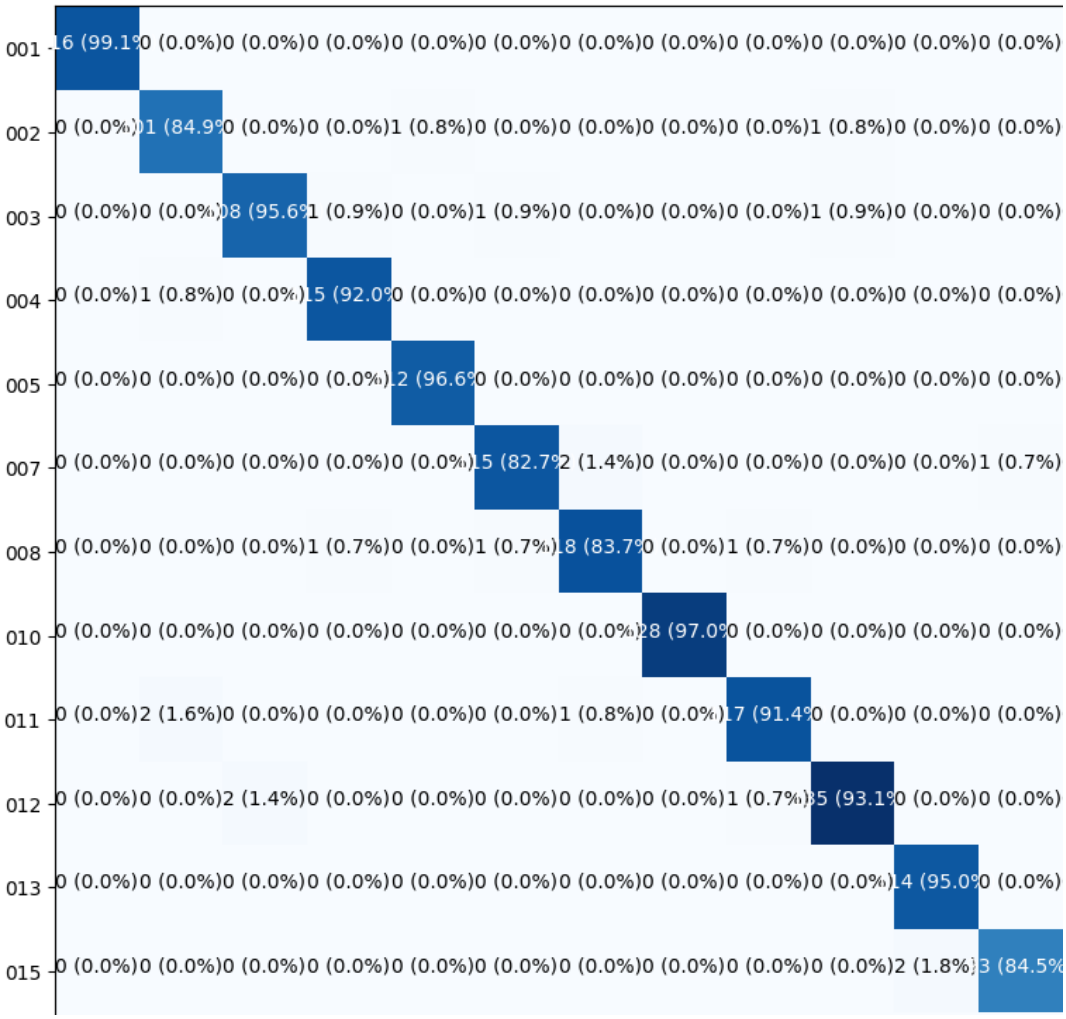


Рисунок 4.12 – Матриця плутанини для 12 користувачів

Джерело: авторська розробка

Матриця плутанини для усіх 52 користувачів наведено на рисунку 4.13, як видно для усіх класів точність передбачення понад 90%.

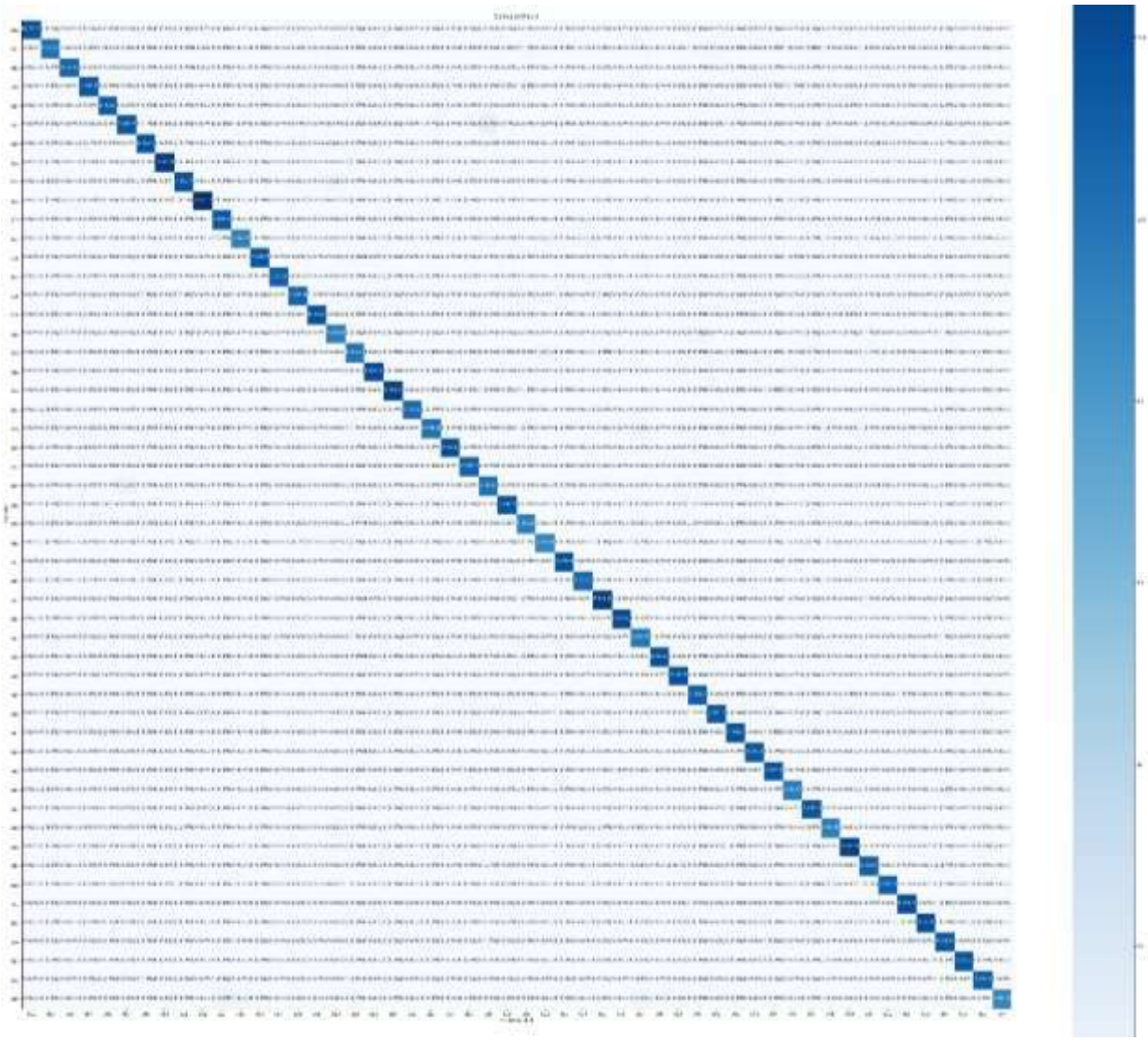


Рисунок 4.13 – Матриця плутанини

Джерело: авторська розробка

4.3 Виклад результатів

За результатами експериментів, accuracy та recall запропонованої моделі становить близько 93,08%, а precision – 93,15%. Ці значення свідчать про високу здатність моделі до правильної класифікації користувачів за їх KD.

Значення F1-міри, яке дорівнює близько 93,05%, також вказує на високу якість класифікації. Ця метрика враховує як accuracy, так і precision, що є важливим для задач, де можуть виникати перекоси у розподілі класів.

Значення площі під ROC-кривою для моделі наближається до 1, що свідчить про високу здатність розрізняти між позитивним та негативним класами.

Детальний аналіз матриці плутанини показав точність класифікації для всіх 52 класів користувачів понад 90%. Це дозволяє впевнено стверджувати, що модель добре справляється з класифікацією на рівні окремих користувачів.

Загалом, запропонована модель нейромережевої аутентифікації користувачів за KD демонструє більшу точність, ніж існуючі моделі, що описані в попередніх розділах роботи. Вона може бути якісним інструментом для підвищення рівня захисту систем та дотримання приватності користувачів.

4.4 Висновок за розділом

В результаті проведених експериментів, можна зробити висновок, що запропонована модель нейромережевої аутентифікації користувачів за допомогою KD демонструє гарну точність в класифікації користувачів. Оцінка моделі на основі ключових метрик таких як accuracy, precision, recall та f1-score підтверджує, що модель впоралася із завданням аутентифікації на високому рівні точності.

Значення площі під ROC-кривою також підтверджують високу здатність моделі розрізняти між позитивними та негативними класами. Детальний аналіз матриці плутанини показав, що модель досягає точності класифікації понад 90% для всіх 52 класів користувачів, що свідчить про її здатність ідентифікувати користувачів із високим рівнем надійності.

Усі ці результати дозволяють впевнено стверджувати, що запропонована модель може бути якісним інструментом для підвищення рівня захисту систем та збереження приватності користувачів. Ця модель має потенціал для застосування в різних областях, де важлива аутентифікація користувачів на високому рівні безпеки.

ВИСНОВОК

Дана робота присвячена розробці нейронної мережі для аутентифікації користувачів за клавіатурним почерком.

У розділі 1 оглянуто та порівняно різні методи аутентифікації за клавіатурним почерком, включаючи статистичні, динамічні та використання прийняття рішень на основі HMMs. Виявлено, що застосування нейромережевих підходів має потенціал, але вимагає уваги до проблем, таких як обсяг доступних даних та точність при роботі з великими обсягами інформації.

У розділі 2 представлено архітектуру нейромережі для аутентифікації користувача на основі клавіатурного почерку. Використання згорткових та рекурентних шарів дозволило розробити модель, що розпізнає та аутентифікує користувачів з високою точністю.

У розділі 3 проведено аналіз набору даних, що підкреслило різноманітність та важливість певних параметрів для аутентифікації користувача за клавіатурним почерком. Це дозволило ідентифікувати ключові аспекти, необхідні для подальшого аналізу та розробки моделі.

У розділі 4 представлені результати експериментів, які підтвердили високу точність запропонованої моделі. Модель досягла високих значень ключових метрик, підтверджуючи її здатність якісно класифікувати та ідентифікувати користувачів.

У цілому, дослідження показує, що нейромережева аутентифікація користувачів за клавіатурним почерком має великий потенціал у сфері захисту даних та ідентифікації користувачів. Запропонована модель може бути використана для підвищення рівня безпеки систем та збереження приватності користувачів в різних сферах. Такі дослідження сприяють розвитку технологій безпеки та можуть бути використані для майбутніх застосувань у різних галузях, де важлива надійна аутентифікація користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Giot R., Rocha A. Siamese networks for static keystroke dynamics authentication. 2019 IEEE international workshop on information forensics and security (WIFS), Delft, Netherlands, 9–12 December 2019. 2019. URL: https://doi.org/10.1109/wifs47025.2019.9035100 (date of access: 18.09.2023).

2. Kochegurova E., Luneva E., Gorokhova E. On continuous user authentication via hidden free-text based monitoring. Advances in intelligent systems and computing. Cham, 2018. P. 66–75. URL: https://doi.org/10.1007/978-3-030-01821-4_8 (date of access: 18.09.2023).

3. Deng Y., Zhong Y. Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets. ISRN signal processing. 2013. Vol. 2013. P. 1–7. URL: https://doi.org/10.1155/2013/565183 (date of access: 18.09.2023).

4. Sahu C., Banavar M. A nonlinear feature transformation-based multi-user classification algorithm for keystroke dynamics. 2021 55th asilomar conference on signals, systems, and computers, Pacific Grove, CA, USA, 31 October – 3 November 2021. 2021. URL: https://doi.org/10.1109/ieeeconf53345.2021.9723223 (date of access: 19.09.2023).

5. Multi-View discriminant analysis / M. Kan et al. IEEE transactions on pattern analysis and machine intelligence. 2016. Vol. 38, no. 1. P. 188–194. URL: https://doi.org/10.1109/tpami.2015.2435740 (date of access: 07.07.2023).

6. Keystroke biometric user verification using hidden markov model / M. L. Ali et al. 2016 IEEE 3rd international conference on cyber security and cloud computing (cscloud), Beijing, China, 25–27 June 2016. 2016. URL: https://doi.org/10.1109/cscloud.2016.23 (date of access: 19.09.2023).

7. Chaudhuri C., Chaudhuri A., Khan A. Authentication of offline signatures based on central tendency of features and dynamic time warping values preserved for genuine cases. 2014 fourth international conference of emerging

applications of information technology (EAIT), Kolkata, India, 19–21 December 2014. 2014. URL: https://doi.org/10.1109/eait.2014.38 (date of access: 19.09.2023).

8. Alghamdi S. J., Elrefaei L. A. Dynamic user verification using touch keystroke based on medians vector proximity. 2015 7th international conference on computational intelligence, communication systems and networks (cicsyn), Riga, Latvia, 3–5 June 2015. 2015. URL: https://doi.org/10.1109/cicsyn.2015.31 (date of access: 19.09.2023).

9. Ali M. L., Thakur K., C. Tappert C. User authentication and identification using neural network. I-manager's journal on pattern recognition. 2015. Vol. 2, no. 2. P. 34–45. URL: https://doi.org/10.26634/jpr.2.2.3567 (date of access: 19.09.2023).

10. Towards keystroke analysis using neural network for multi-factor authentication of learner recognition in on-line examination. Multi-Disciplinary manila (philippines) conferences jan. 23-24, 2017, manila (philippines). 2017. URL: https://doi.org/10.17758/uruae.ae0117514 (date of access: 19.09.2023).

11. Continuous user identification via touch and movement behavioral biometrics / C. Bo et al. 2014 IEEE international performance computing and communications conference (IPCCC), Austin, TX, USA, 5–7 December 2014. 2014. URL: https://doi.org/10.1109/pccc.2014.7017067 (date of access: 19.09.2023).

12. Banerjee S. P., Woodard D. Biometric authentication and identification using keystroke dynamics: a survey. Journal of pattern recognition research. 2012. Vol. 7, no. 1. P. 116–139. URL: https://doi.org/10.13176/11.427 (date of access: 19.09.2023).

13. Casey K. Using keystroke analytics to improve pass-fail classifiers. Journal of learning analytics. 2017. Vol. 4, no. 2. URL: https://doi.org/10.18608/jla.2017.42.14 (date of access: 19.09.2023).

14. Neural networks: an overview of early research, current frameworks and new challenges / A. Prieto et al. Neurocomputing. 2016. Vol. 214. P. 242–268. URL: https://doi.org/10.1016/j.neucom.2016.06.014 (date of access: 19.09.2023).

15. Learning activation functions in deep (spline) neural networks / P. Bohra et al. IEEE open journal of signal processing. 2020. Vol. 1. P. 295–309. URL: https://doi.org/10.1109/ojsp.2020.3039379 (date of access: 19.09.2023).

16. Sharma S., Sharma S., Athaiya A. Activation functions in neural networks. International journal of engineering applied sciences and technology. 2020. Vol. 04, no. 12. P. 310–316. URL: https://doi.org/10.33564/ijeast.2020.v04i12.054 (date of access: 19.09.2023).

17. Ezeafulukwe, Uzoamaka A.; Darus, Maslina; Fadipe-Joseph, Olubunmi Abidemi. On analytic properties of a sigmoid function. International Journal of Mathematics & Computer Science . 2018, Vol. 13 Issue 2, p171-178. 8p.

18. De Ryck T., Lanthaler S., Mishra S. On the approximation of functions by tanh neural networks. Neural networks. 2021. Vol. 143. P. 732–750. URL: https://doi.org/10.1016/j.neunet.2021.08.015 (date of access: 19.09.2023).

19. Self-gated rectified linear unit for performance improvement of deep neural networks / I. Jahan et al. ICT express. 2022. URL: https://doi.org/10.1016/j.icte.2021.12.012 (date of access: 19.09.2023).

20. Deep speaker embedding extraction with channel-wise feature responses and additive supervision softmax loss function / J. Zhou et al. Interspeech 2019. ISCA, 2019. URL: https://doi.org/10.21437/interspeech.2019-1704 (date of access: 19.09.2023).

21. AutoLoss: automated loss function search in recommendations / X. Zhao et al. KDD '21: the 27th ACM SIGKDD conference on knowledge discovery and data mining, Virtual Event Singapore. New York, NY, USA, 2021. URL: https://doi.org/10.1145/3447548.3467208 (date of access: 19.09.2023).

22. Ho Y., Wookey S. The real-world-weight cross-entropy loss function: modeling the costs of mislabeling. IEEE access. 2020. Vol. 8. P. 4806–4813. URL: https://doi.org/10.1109/access.2019.2962617 (date of access: 19.09.2023).

23. Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N.A. (2020). Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. ArXiv, abs/2002.06305.

24. Jais I. K. M., Ismail A. R., Nisa S. Q. Adam optimization algorithm for wide and deep neural network. Knowledge engineering and data science. 2019. Vol. 2, no. 1. P. 41. URL: https://doi.org/10.17977/um018v2i12019p41-46 (date of access: 19.09.2023).

25. Kingma, D.P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. CoRR, abs/1412.6980.

26. Initial margin simulation with deep learning / X. Ma et al. SSRN electronic journal. 2019. URL: https://doi.org/10.2139/ssrn.3357626 (date of access: 19.09.2023).

27. Eliasy A., Przychodzen J. The role of AI in capital structure to enhance corporate funding strategies. Array. 2020. Vol. 6. P. 100017. URL: https://doi.org/10.1016/j.array.2020.100017 (date of access: 19.09.2023).

28. Аналіз моделей рекурентних нейронних мереж для розробки інтелектуальних чат-ботів [Текст] / А. Яровий, О. Прозор, Ю. Щипський // Proceedings of the XII International scientific-practical conference«INTERNET-EDUCATION-SCIENCE» (IES-2020), Ukraine, Vinnytsia, 26-29 May 2020. – Vinnytsia : VNTU, 2020. – С. 18–21.

29. O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

30. Goutte C., Gaussier E. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. Lecture notes in computer science. Berlin, Heidelberg, 2005. P. 345–359. URL: https://doi.org/10.1007/978-3-540-31865-1_25 (date of access: 19.09.2023).

31. Employing long short-term memory and Facebook prophet model in air temperature forecasting / T. Toharudin et al. Communications in statistics - simulation and computation. 2021. P. 1–24. URL: https://doi.org/10.1080/03610918.2020.1854302 (date of access: 19.09.2023).

32. Implementation of long short-term memory (LSTM) for user authentication based on keystroke dynamics / A. Ferhatovic et al. Southeast europe

journal of soft computing. 2020. Vol. 9, no. 1. URL: https://doi.org/10.21533/scjournal.v9i1.187 (date of access: 19.09.2023).

33. Deng Y., Zhong Y. Keystroke dynamics advances for mobile devices using deep neural network. Gate to computer science and research. 2015. P. 59–70. URL: https://doi.org/10.15579/gcsr.vol2.ch4 (date of access: 19.09.2023).

34. Varsamopoulos S., Bertels K., Almudever C. G. Comparing neural network based decoders for the surface code. IEEE transactions on computers. 2020. Vol. 69, no. 2. P. 300–311. URL: https://doi.org/10.1109/tc.2019.2948612 (date of access: 19.09.2023).

35. Types of Neural Networks and Definition of Neural Network. _Great Learning Blog: Free Resources what Matters to shape your Career!_. URL: https://www.mygreatlearning.com/blog/types-of-neural-networks/ (date of access: 24.02.2023). Режим доступу: https://www.mygreatlearning.com/blog/types-of-neural-networks/

36. Liu M., Guan J. User keystroke authentication based on convolutional neural network. Communications in computer and information science. Singapore, 2018. P. 157–168. URL: https://doi.org/10.1007/978-981-13-3732-1_13 (date of access: 19.09.2023).

37. Kim J., Kim H., Kang P. Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection. Applied soft computing. 2018. Vol. 62. P. 1077–1087. URL: https://doi.org/10.1016/j.asoc.2017.09.045 (date of access: 19.09.2023).

38. On neural networks for biometric authentication based on keystroke dynamics. Sensors and materials. 2018. P. 385. URL: https://doi.org/10.18494/sam.2018.1757 (date of access: 19.09.2023).

39. Can we use split learning on 1D CNN models for privacy preserving training? / S. Abuadbba et al. ASIA CCS '20: the 15th ACM asia conference on computer and communications security, Taipei Taiwan. New York, NY, USA, 2020. URL: https://doi.org/10.1145/3320269.3384740 (date of access: 19.09.2023).

40. Xiaofeng L., Shengfei Z., Shengwei Y. Continuous authentication by free-text keystroke based on CNN plus RNN. Procedia computer science. 2019. Vol. 147. P. 314–318. URL: https://doi.org/10.1016/j.procs.2019.01.270 (date of access: 19.09.2023).

41. Попередня обробка та аналіз даних: лабораторний практикум для студ. спеціальності 113 «Прикладна математика» / Уклад.: Н. Е. Кондрук. Ужгород: УжНУ, 2023. – 41 с.

42. Comparing Anomaly-Detection Algorithms for Keystroke Dynamics [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: http://www.cs.cmu.edu/~keystroke/#sec1.

43. Killourhy K. S., Maxion R. A. Comparing anomaly-detection algorithms for keystroke dynamics. Networks (DSN), Lisbon, Portugal, 29 June – 2 July 2009. 2009. URL: https://doi.org/10.1109/dsn.2009.5270346 (date of access: 19.09.2023).

44. RandomSearch Tuner [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/api/keras_tuner/tuners/random/

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
ПЗ		Оформлення відповідає ДСТУ 3008:2015

Додаток Б

Фрагмент лістингу програмного коду

```
pip install keras-tuner
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from tensorflow.keras.layers import Input, Dense, Lambda, LSTM,
Bidirectional, Multiply, Activation, Flatten, Dropout, Conv1D,
MaxPooling1D
import keras_tuner
from tensorflow import keras
from kerastuner import RandomSearch
from kerastuner.engine.hyperparameters import HyperParameters
from sklearn.metrics import classification_report

from google.colab import drive
drive.mount('/content/drive')
data = pd.read_csv("/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/DSL-StrongPasswordData.csv")
data.drop(columns=['sessionIndex','rep'],axis=1,inplace=True)
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])],remain
der='passthrough')
unique_cols=data['subject'].nunique()

data=np.array(ct.fit_transform(data))
X=data[:,unique_cols:]
y=data[:,:unique_cols]
print(y.shape)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

```
X.shape
y.shape
```

```
def build_model(hp):
```

```
  model=tf.keras.models.Sequential()
  model.add(tf.keras.layers.Conv1D(filters=hp.Int('conv_1_filter',
min_value=16, max_value=128, step=16),
                                   kernel_size=hp.Choice('conv_1_kernel',
values = [2,3]),
                                   activation=hp.Choice('conv_1_activation
', values = ['relu', 'tanh', 'sigmoid']),
                                   input_shape=(X_train.shape[1], 1)))

  model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

  model.add(tf.keras.layers.Conv1D(filters=hp.Int('conv_2_filter',
min_value=16, max_value=128, step=16),
                                   kernel_size=hp.Choice('conv_2_kernel',
values = [2,3]),
                                   activation=hp.Choice('conv_2_activation
', values = ['relu', 'tanh', 'sigmoid'])))

  model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

  model.add(tf.keras.layers.LSTM(units=hp.Int('LSTM_units', min_value=16,
max_value=128, step=16), return_sequences=True))

  model.add(tf.keras.layers.Flatten())

  model.add(tf.keras.layers.Dense(units=hp.Int('dense_1_units',
min_value=16, max_value=128, step=16),
                                  activation=hp.Choice('dense_activation',
values = ['relu', 'tanh'])))

  model.add(Dropout(rate = hp.Float('dropout_rate', min_value=0.1,
max_value=0.9, step=0.1)))

  model.add(tf.keras.layers.Dense(units=unique_cols,activation='softmax'))

  model.compile(optimizer= 'adam',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
  return model
```

```
tuner = keras_tuner.RandomSearch(
    build_model,
    objective='accuracy',
    max_trials=50,
    overwrite = True)
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
```

```
tuner.search(X_train, y_train, epochs=100, validation_split=0.1, callbacks
= [callback])
tuner.results_summary()
```

```
model=tf.keras.models.Sequential()

model.add(tf.keras.layers.Conv1D(64,2, activation='relu',
input_shape=(X_train.shape[1], 1)))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.Conv1D(112,2, activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.LSTM(32, return_sequences=True))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(80, activation='tanh'))

model.add(Dropout(0.1))

model.add(tf.keras.layers.Dense(units=unique_cols,activation='softmax'))

model.compile(optimizer= 'adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_8=model.fit(X_train, y_train, epochs = 100, batch_size=8,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_8,acc_8 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_8)+" accuracy: "+str(acc_8))
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_16=model.fit(X_train, y_train, epochs = 100, batch_size=16,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_16,acc_16 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_16)+" accuracy: "+str(acc_16))
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_32=model.fit(X_train, y_train, epochs = 100, batch_size=32,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_32,acc_32 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_32)+" accuracy: "+str(acc_32))
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_48=model.fit(X_train, y_train, epochs = 100, batch_size=48,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_48,acc_48 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_48)+" accuracy: "+str(acc_48))
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_64=model.fit(X_train, y_train, epochs = 100, batch_size=64,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_64,acc_64 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_64)+" accuracy: "+str(acc_64))
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_128=model.fit(X_train, y_train, epochs = 100, batch_size=128,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_128,acc_128 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_128)+" accuracy: "+str(acc_128))
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist_256=model.fit(X_train, y_train, epochs = 100, batch_size=256,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss_256,acc_256 = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss_256)+" accuracy: "+str(acc_256))
```

```
plt.figure()
plt.plot([8, 16, 32, 48, 64, 128,256], [acc_8, acc_16, acc_32, acc_48,
acc_64, acc_128, acc_256])
plt.scatter([8, 16, 32, 48, 64, 128, 256], [acc_8, acc_16, acc_32, acc_48,
acc_64, acc_128, acc_256])
plt.xlabel("Розмір партії")
```

```
plt.ylabel("Точність")
plt.grid()
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/graf_acc')
plt.show()
```

```
model=tf.keras.models.Sequential()

model.add(tf.keras.layers.Conv1D(64,2, activation='relu',
input_shape=(X_train.shape[1], 1)))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.Conv1D(112,2, activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.LSTM(32, return_sequences=True))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(80, activation='tanh'))

model.add(Dropout(0.1))

model.add(tf.keras.layers.Dense(units=unique_cols,activation='softmax'))

model.compile(optimizer= 'adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
hist=model.fit(X_train, y_train, epochs = 100, batch_size=128,
validation_split = 0.1, verbose = 1)
```

```
loss,acc = model.evaluate(X_test, y_test, verbose = 1)
print("loss: "+str(loss)+" accuracy: "+str(acc))
```

```
plt.plot(hist.history['loss'], 'b-', label='train loss')
plt.plot(hist.history['val_loss'], 'r-', label='val loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='lower right')
plt.grid()
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/graf_epoch_loss')
plt.show()

plt.plot(hist.history['accuracy'], 'b-', label='train accuracy')
plt.plot(hist.history['val_accuracy'], 'r-', label='val accuracy')
plt.ylabel('Accuracy')
```

```
plt.xlabel('Epoch')
plt.grid()
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/graf_epoch_acc')
plt.show()
```

```
model1=tf.keras.models.Sequential()

model1.add(tf.keras.layers.Conv1D(64,2, activation='relu',
input_shape=(X_train.shape[1], 1)))

model1.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model1.add(tf.keras.layers.Conv1D(112,2, activation='relu'))
model1.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model1.add(tf.keras.layers.LSTM(32, return_sequences=True))

model1.add(tf.keras.layers.Flatten())

model1.add(tf.keras.layers.Dense(80, activation='tanh'))

model1.add(Dropout(0.1))

model1.add(tf.keras.layers.Dense(units=unique_cols,activation='softmax'))

model1.compile(optimizer= 'adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
callback=tf.keras.callbacks.EarlyStopping(monitor='accuracy',mode='auto',p
atience=5,verbose=1,restore_best_weights=True)
hist=model1.fit(X_train, y_train, epochs = 100, batch_size=128,
validation_split = 0.1, verbose = 1, callbacks = [callback])
```

```
loss,acc = model1.evaluate(X_test, y_test, verbose = 1)
print("loss_test: "+str(loss)+" accuracy_test: "+str(acc))
```

```
loss,acc = model1.evaluate(X_train, y_train, verbose = 1)
print("loss_train: "+str(loss)+" accuracy_train: "+str(acc))
```

```
y_pred = model1.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_test, axis=1)
report = classification_report(y_true_classes, y_pred_classes)
print(report)
```

```
plt.plot(hist.history['loss'], 'b-', label='train loss')
plt.plot(hist.history['val_loss'], 'r-', label='val loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='lower right')
plt.grid()
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/graf_epoch_loss_cal')
plt.show()

plt.plot(hist.history['accuracy'], 'b-', label='train accuracy')
plt.plot(hist.history['val_accuracy'], 'r-', label='val accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.grid()
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/graf_epoch_acc_cal')
plt.show()
```

```
tf.keras.saving.save_model(model1, '/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/model.h5')
```

```
print(model1.summary())
```

```
img_file = '/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/model_arch.png'

tf.keras.utils.plot_model(model1, to_file=img_file, show_shapes=True,
show_layer_names=True)
```

```
evaluate = model1.evaluate(X_test, y_test)
y_probs = model1.predict(X_test)
y_preds = y_probs.argmax(axis=1)
y_test_preds = y_test.argmax(axis=1)

print(evaluate)
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score, f1_score, precision_score

accuracy = accuracy_score(y_test_preds, y_preds)
precision = precision_score(y_test_preds, y_preds, average='weighted')
recall = recall_score(y_test_preds, y_preds, average='weighted')
f1score = f1_score(y_test_preds, y_preds, average='weighted')

print(f"Accuracy = {accuracy}")
print(f"Precision = {precision}")
print(f"Recall = {recall}")
```

```
print(f"F1 Score = {f1score}")
```

```
plt.figure()
plt.plot(["Accuracy", "Precision", "Recall", "F1 Score"], [accuracy,
precision, recall, f1score])
plt.scatter(["Accuracy", "Precision", "Recall", "F1 Score"], [accuracy,
precision, recall, f1score])
plt.xlabel("Параметри оцінки")
plt.ylabel("Значення")
plt.grid()
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/eval')
plt.show()
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

# Перетворення міток в однокодовий формат для розрахунку ROC AUC
y_test_bin = label_binarize(y_test, classes=range(unique_cols))

# Прогнозування ймовірностей для кожного класу
y_probs = model1.predict(X_test)

# Обчислення ROC-кривої та ROC-площі для кожного класу
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(unique_cols):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Побудова ROC-кривої для кожного класу
plt.figure(figsize=(6, 6))
for i in range(unique_cols):
    plt.plot(fpr[i], tpr[i], lw=2, label=f'Користувач {i+1} (AUC =
{roc_auc[i]:.4f})')

plt.plot([0, 1.0], [0, 1.0], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('Частота хибнопозитивних результатів')
plt.ylabel('Частота істинних позитивних результатів')
plt.title('Крива ROC AUC')
#plt.legend(loc='best')
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/roc')
plt.show()
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve, auc

precision = dict()
recall = dict()
pr_auc = dict()

for i in range(unique_cols):
    precision[i], recall[i], _ = precision_recall_curve(y_test_bin[:, i],
y_probs[:, i])
    pr_auc[i] = auc(recall[i], precision[i])

precision["micro"], recall["micro"], _ =
precision_recall_curve(y_test_bin.ravel(), y_probs.ravel())
pr_auc["micro"] = auc(recall["micro"], precision["micro"])

plt.figure(figsize=(15, 15))
for i in range(unique_cols):
    plt.plot(recall[i], precision[i], lw=2, label=f'Користувач {i} (AUC =
{pr_auc[i]:.2f})')

plt.plot(recall["micro"], precision["micro"], color='gold', lw=2,
linestyle='--',
         label=f'Середнє (AUC = {pr_auc["micro"]:.2f})')

plt.xlabel('Частота хибнопозитивних результатів')
plt.ylabel('Частота істинних позитивних результаів')
plt.title('Значення AUC')
plt.legend(loc='best')
plt.grid(True)
plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/AUC')
plt.show()
```

```
import itertools
from sklearn.metrics import confusion_matrix

def make_confusion_matrix(y_true, y_pred, classes=None, figsize=(8, 8),
text_size=15):

  cm = confusion_matrix(y_true, y_pred)
  cm_norm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]
  n_classes = cm.shape[0]

  fig, ax = plt.subplots(figsize=figsize)
  cax = ax.matshow(cm, cmap=plt.cm.Blues)
  fig.colorbar(cax)
```

```
  if classes:
    labels = classes
  else:
    labels = np.arange(cm.shape[0])


  ax.set(title="Confusion Matrix",
         xlabel="Predicted label",
         ylabel="True label",
         xticks=np.arange(n_classes),
         yticks=np.arange(n_classes),
         xticklabels=labels,
         yticklabels=labels)


  ax.xaxis.set_label_position("bottom")
  ax.xaxis.tick_bottom()


  threshold = (cm.max() + cm.min()) / 2.


  for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, f"{cm[i, j]} ({cm_norm[i, j]*100:.1f}%)",
             horizontalalignment="center",
             color="white" if cm[i, j] > threshold else "black",
             size=text_size)

  plt.savefig('/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/confusion_matrix.png', bbox_inches='tight')
```

```
classes = ['001', '002', '003', '004', "005", "007" , "008", "010",
           "011", "012", "013", "015", "016", "017", "018", "019", "020",
           "021", "022", "024", "025", "026", "027", "028", "029", "030",
           "031", "032", "033", "034", "035", "036", "037", "038", "039",
"040",
           "041", "042", "043", "044", "046", "047", "048", "049", "050",
           "051", "052", "053", "054", "055", "056", "057"]


make_confusion_matrix(y_true=y_test_preds,
                      y_pred=y_preds,
                      classes=classes,
                      figsize=(50, 50),
                      text_size=10)
```

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

```
import math
x = np.linspace(-10, 10, 100)
y = 1/(1 + np.exp(-x))

plt.plot(x, y, label="Sigmoid")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Графік сигмоїдальної функції ")
plt.grid(True)
plt.legend()
plt.savefig('Sigmoid.png')
plt.show()
```

```
x = np.linspace(-10, 10, 100)
y = 2/(1 + np.exp(-2*x))-1

plt.plot(x, y, label="tanH")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Графік функції гіперболічного тангенса")
plt.grid(True)
plt.legend()
plt.savefig('tanH.png')
plt.show()
```

```
x = np.linspace(-10, 10, 100)
y = np.maximum(0, x)

plt.plot(x, y, label="ReLU")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Графік функції зрізаного лінійного вузла")
plt.grid(True)
plt.legend()
plt.savefig('ReLU.png')
plt.show()
```

```
def softmax(x):
    exp_x = np.exp(x)
    sum_exp_x = np.sum(exp_x)
    softmax_x = exp_x / sum_exp_x
    return softmax_x

x = np.linspace(-10, 10, 100)
plt.plot(x, softmax(x), label="Softmax")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Графік багатозмінної логістичної функції")
```

```
plt.grid(True)
plt.legend()
plt.savefig('Softmax.png')
plt.show()
```

```
x = np.linspace(-10, 10, 100)

y1 = 1/(1 + np.exp(-x))
plt.plot(x, y1, label="Sigmoid")

y2 = 2/(1 + np.exp(-2*x))-1
plt.plot(x, y2, label="tanH")

y3 = np.maximum(0, x)
plt.plot(x, y3, label="ReLU ")

def softmax(x):
    exp_x = np.exp(x)
    sum_exp_x = np.sum(exp_x)
    softmax_x = exp_x / sum_exp_x
    return softmax_x

plt.plot(x, softmax(x), label="Softmax")

plt.xlabel("x")
plt.ylabel("f(x)")
plt.grid(True)
plt.legend()
plt.savefig('All.png')
plt.show()
```

```
def binary_cross_entropy(y_true, y_pred):
    epsilon = 1e-15
    y_pred = np.clip(y_pred, epsilon, 1 - epsilon)
    return - (y_true * np.log(y_pred) + (1 - y_true) * np.log(1 - y_pred))

y_pred = np.linspace(0.01, 0.99, 100)

# For y_true = 0
y_true_0 = np.zeros_like(y_pred)
loss_values_0 = binary_cross_entropy(y_true_0, y_pred)

# For y_true = 1
y_true_1 = np.ones_like(y_pred)
loss_values_1 = binary_cross_entropy(y_true_1, y_pred)

plt.plot(y_pred, loss_values_0, label="y = 0")
plt.plot(y_pred, loss_values_1, label="y = 1")
plt.xlabel("Прогнозована ймовірність")
```

```
plt.ylabel("Бінарна перехресна втрата ентропії")
plt.title("Графік бінарної функції втрати перехресної ентропії")
plt.legend()
plt.grid(True)
plt.savefig('log loss.png')

plt.show()
```

```
def categorical_cross_entropy(y_true, y_pred):
    epsilon = 1e-15
    y_pred = np.clip(y_pred, epsilon, 1 - epsilon)
    return - np.sum(y_true * np.log(y_pred))

num_classes = 5
y_true = np.array([0, 0, 1, 0, 0])
y_pred = np.linspace(0.01, 0.99, 100)

loss_values = [categorical_cross_entropy(y_true, [p] * num_classes) for p
in y_pred]

plt.plot(y_pred, loss_values)
plt.xlabel("Прогнозована ймовірність для істинного класу")
plt.ylabel("Категоріальна перехресна втрата ентропії")
plt.title("Графік категоріальної функції втрати перехресної ентропії")
plt.grid(True)
plt.savefig('softmaxloss.png')
plt.show()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import seaborn as sn

plt.style.use('ggplot')
#plt.style.use('dark_background')

%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv("/content/drive/Othercomputers/Мій
ноутбук/Магістратура/Диплом/DSL-StrongPasswordData.csv")
```

```
df.shape
df.head()
```

```
df.info()
df.describe()
```

```
df[[x for x in df.columns if 'H' in x]].describe()
```

```
df[[x for x in df.columns if 'DD' in x]].describe()
```

```
df[[x for x in df.columns if 'UD' in x]].describe()
```

```
def load_data():
    H_columns  = [col for col in df.columns if col.startswith('H')]
    DD_columns = [col for col in df.columns if col.startswith('DD')]
    UD_columns = [col for col in df.columns if col.startswith('UD')]

    data = {}
    data['total'] = df.drop(columns=['subject', 'sessionIndex', 'rep'])
    data['H']     = df[H_columns]
    data['DD']    = df[DD_columns]
    data['UD']    = df[UD_columns]
    data['pca3']  =
pd.DataFrame(PCA(n_components=3).fit_transform(data['total']))
    data['pca10'] =
pd.DataFrame(PCA(n_components=10).fit_transform(data['total']))

    return data, df['subject'].values
```

```
data, y = load_data()

# One hot encoding
Y = pd.get_dummies(y).values
n_classes = Y.shape[1]
```

```
df_plot = data['DD'].copy()

plt.figure(figsize=(16, 9))
for i, y0 in enumerate(np.unique(y)[1:7]):
    plt.subplot(2, 3, i + 1)
    for x in df_plot[y == y0].values:
        plt.plot(x, color='k', alpha=0.1)
    plt.title('DD для {}'.format(y0))
    plt.xlabel('Індекс ключа')
    plt.ylabel('Тривалість')
plt.tight_layout()
```

```
df_plot = data['DD'].copy()
```

```
plt.figure(figsize=(16, 9))
for i, y0 in enumerate(np.unique(y)[:1]):
    plt.subplot(2, 3, i + 1)
    for x in df_plot[y == y0].values:
        plt.plot(x, color='k', alpha=0.1)
    plt.title('DD для авторського вводу')
    plt.xlabel('Індекс ключа')
    plt.ylabel('Тривалість')
plt.tight_layout()
```

```
df_plot = data['DD'].copy()
df_plot['subject'] = y
df_plot = df_plot.groupby('subject').mean()
df_plot.iloc[:6].T.plot(figsize=(16, 9), title='Середній DD')
plt.xlabel('Індекс ключа')
plt.ylabel('Тривалість');
```

```
df_plot = data['H'].copy()

plt.figure(figsize=(16, 9))
for i, y0 in enumerate(np.unique(y)[1:7]):
    plt.subplot(2, 3, i + 1)
    for x in df_plot[y == y0].values:
        plt.plot(x, color='k', alpha=0.1)
    plt.title('H для {}'.format(y0))
    plt.xlabel('Індекс ключа')
    plt.ylabel('Тривалість')
plt.tight_layout()
```

```
df_plot = data['H'].copy()

plt.figure(figsize=(16, 9))
for i, y0 in enumerate(np.unique(y)[:1]):
    plt.subplot(2, 3, i + 1)
    for x in df_plot[y == y0].values:
        plt.plot(x, color='k', alpha=0.1)
    plt.title('H для для авторського вводу')
    plt.xlabel('Індекс ключа')
    plt.ylabel('Тривалість')
plt.tight_layout()
```

```
df_plot = data['H'].copy()
df_plot['subject'] = y
df_plot = df_plot.groupby('subject').mean()
df_plot.iloc[:6].T.plot(figsize=(16, 9), title='Середнє H')
plt.xlabel('Індекс ключа')
plt.ylabel('Тривалість');
```

```
df_plot = data['UD'].copy()


plt.figure(figsize=(16, 9))
for i, y0 in enumerate(np.unique(y)[1:7]):
    plt.subplot(2, 3, i + 1)
    for x in df_plot[y == y0].values:
        plt.plot(x, color='k', alpha=0.1)
    plt.title('UD для {}'.format(y0))
    plt.xlabel('Індекс ключа')
    plt.ylabel('Тривалість')
plt.tight_layout()
```

```
df_plot = data['UD'].copy()

plt.figure(figsize=(16, 9))
for i, y0 in enumerate(np.unique(y)[:1]):
    plt.subplot(2, 3, i + 1)
    for x in df_plot[y == y0].values:
        plt.plot(x, color='k', alpha=0.1)
    plt.title('UD для для авторськогоВ вводу')
    plt.xlabel('Індекс ключа')
    plt.ylabel('Тривалість')
plt.tight_layout()
```

```
df_plot = data['UD'].copy()
df_plot['subject'] = y
df_plot = df_plot.groupby('subject').mean()
df_plot.iloc[:6].T.plot(figsize=(16, 9), title='Середнє UD')
plt.xlabel('Індекс ключа')
plt.ylabel('Тривалість');
```

```
indx_col = ['subject','sessionIndex','rep']
indx = df[indx_col]
#label = [ int(x[1:]) for x in data['subject']]
features = df.drop(columns=indx_col)
```

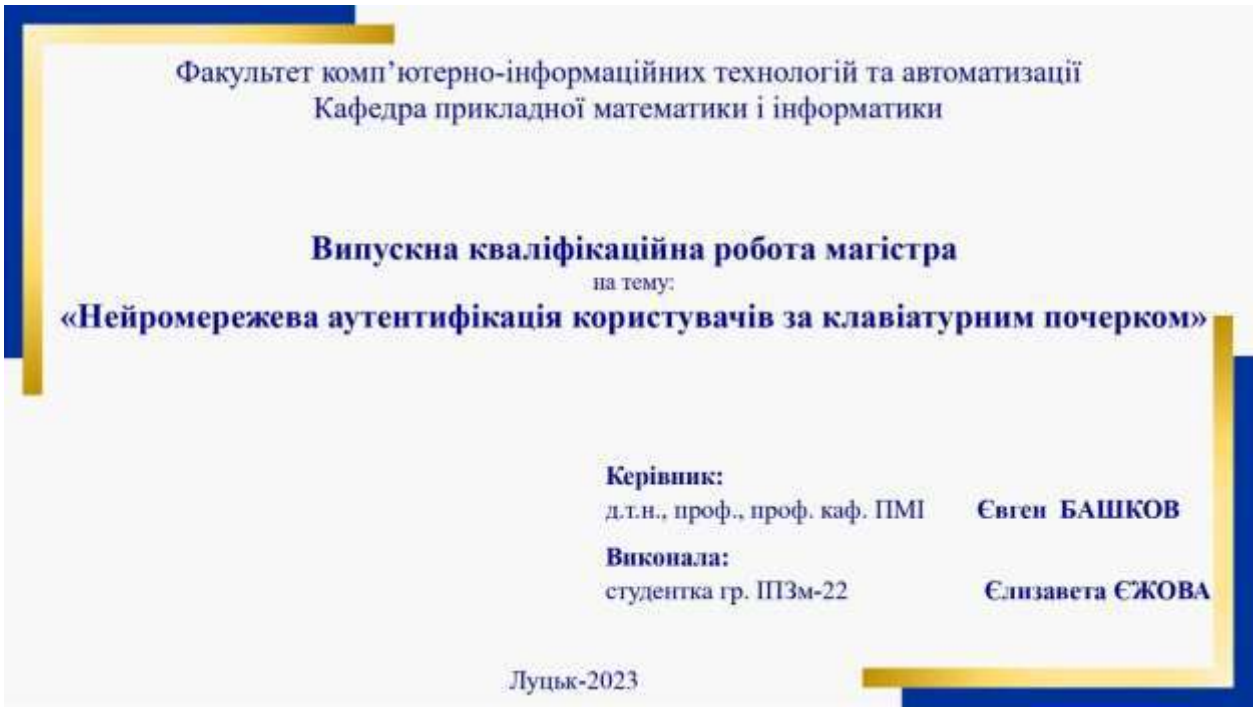
```
corrMatrix = df[['H.period', 'DD.period.t',
       'UD.period.t', 'H.t', 'DD.t.i', 'UD.t.i', 'H.i', 'DD.i.e',
'UD.i.e',
       'H.e', 'DD.e.five', 'UD.e.five', 'H.five', 'DD.five.Shift.r',
       'UD.five.Shift.r', 'H.Shift.r', 'DD.Shift.r.o', 'UD.Shift.r.o',
'H.o',
       'DD.o.a', 'UD.o.a', 'H.a', 'DD.a.n', 'UD.a.n', 'H.n', 'DD.n.l',
       'UD.n.l', 'H.l', 'DD.l.Return', 'UD.l.Return', 'H.Return']].corr()
fig, ax = plt.subplots(figsize=(10,10))
sn.heatmap(corrMatrix, linewidths=0.1, cmap="coolwarm")
plt.show()
```

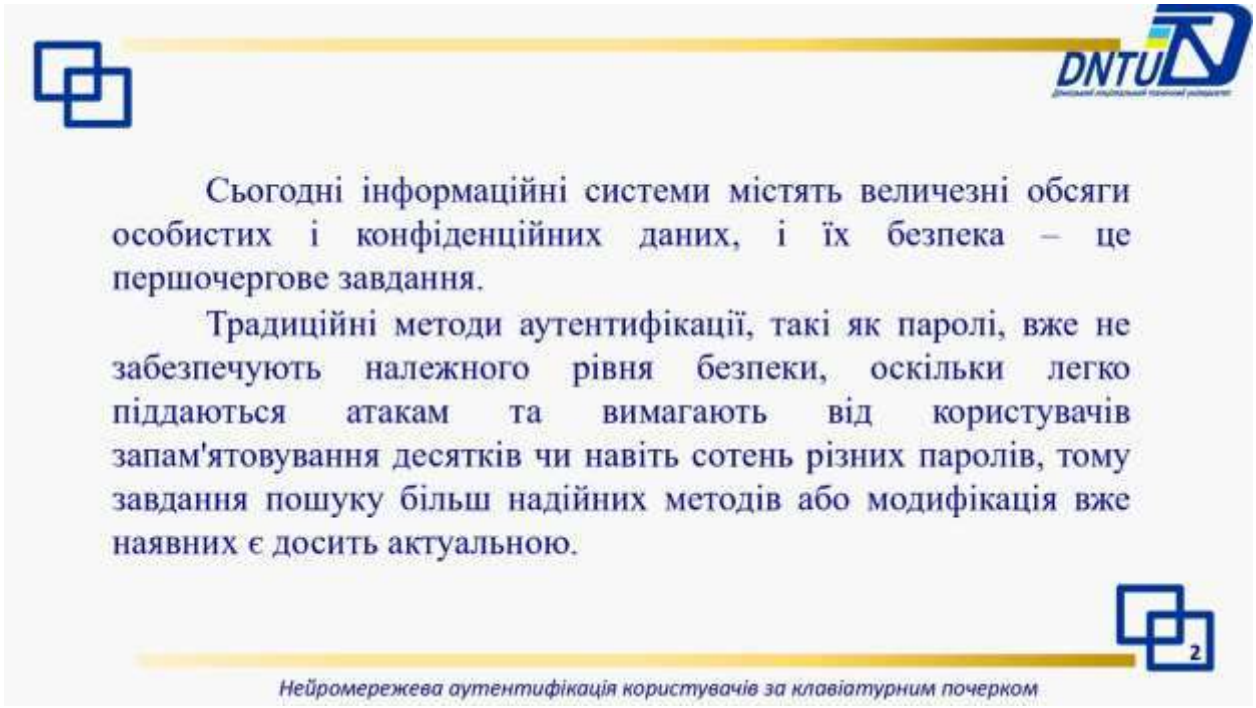
```
features_H_UD = features.drop(columns=[x for x in features.columns if 'DD'
in x])
df = features_H_UD.corr()

fig, ax = plt.subplots(figsize=(10,10))
sn.heatmap(df, linewidths=0.1, cmap="coolwarm")
plt.show()
```

Додаток В

Матеріали презентації



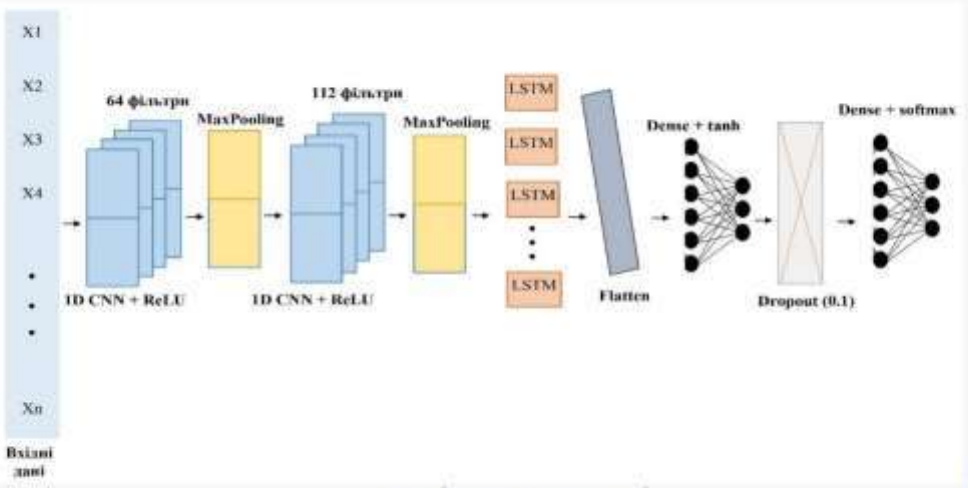


Мета роботи

Підвищення рівня захисту комп'ютерно-інформаційних систем шляхом високоточної автоматичної ідентифікації користувачів за їх KD за допомогою NNs аутентифікації, що забезпечить ідентифікацію користувача на основі унікальних характеристик його введення на клавіатурі.

Задачі роботи

Проведення порівняльного аналізу наявних методів аутентифікації користувачів за KD, зокрема, з використання NNs, для подальшого обрання архітектури, яка ймовірно забезпечить високу точність в завдання аутентифікації за KD, проведення налаштування, навчання та тестування NN, використовуючи відповідні методи оптимізації, щоб досягнути максимальної точності моделі та оцінка точності розробленої архітектури мережі на наборі тестових даних, для оцінки її здатності розрізняти аутентичний почерк від підробленого.



Архітектура мережі

conv1d_input	input:	[(None, 31, 1)]
InputLayer	output:	[(None, 31, 1)]
conv1d	input:	(None, 31, 1)
Conv1D	output:	(None, 30, 64)
max_pooling1d	input:	(None, 30, 64)
MaxPooling1D	output:	(None, 15, 64)
conv1d_1	input:	(None, 15, 64)
Conv1D	output:	(None, 14, 112)
max_pooling1d_1	input:	(None, 14, 112)
MaxPooling1D	output:	(None, 7, 112)

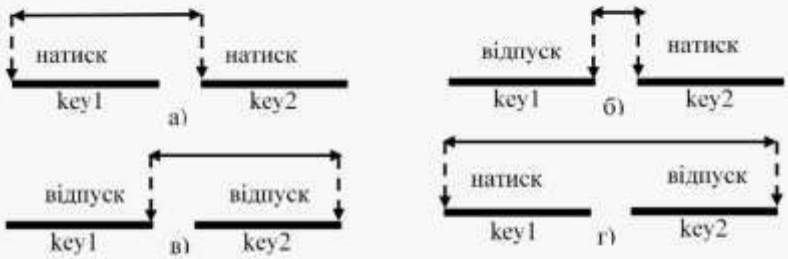
lstm	input:	(None, 7, 112)
LSTM	output:	(None, 7, 32)
flatten	input:	(None, 7, 32)
Flatten	output:	(None, 224)
dense	input:	(None, 224)
Dense	output:	(None, 80)
dropout	input:	(None, 80)
Dropout	output:	(None, 80)
dense_1	input:	(None, 80)
Dense	output:	(None, 52)

Структура запропонованої моделі

Як вхідні дані для навчання NN для аутентифікації користувача обрано датасет, що був зібраний у Carnegie Mellon University для випробування різних моделей розпізнавання, що вміщує інформацію про час натискання клавіш від 51 користувача, кожен з яких вводить пароль .tie5Roanl 400 разів, та додана до нього авторська спроба вводу, той же пароль вводився також протягом 8 сесій по 50 разів у кожній, загалом 400 разів, автору присвоєно ідентифікатор s001.

	A	B	C	D	E	F
1	subject,sessionIndex,rep,H.period,DD.period.t,UD.period.t,H.t,					
2	s002,1,1,0.1491,0.3979,0.2488,0.1069,0.1674,0.0605,0.1169					
3	s002,1,2,0.1111,0.3451,0.2340,0.0694,0.1283,0.0589,0.0908					
4	s002,1,3,0.1328,0.2072,0.0744,0.0731,0.1291,0.0560,0.0821					
5	s002,1,4,0.1291,0.2515,0.1224,0.1059,0.2495,0.1436,0.1040					
6	s002,1,5,0.1249,0.2317,0.1068,0.0895,0.1676,0.0781,0.0903					

Зображення вхідних даних



а) час між натисканням двох кнопок пароля,

б) час між відпусканням попередньої та натисканням наступної кнопки,

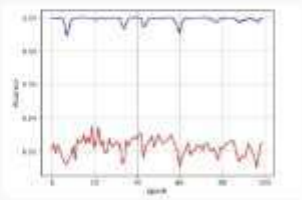
в) час між відпусканням попередньої та наступної кнопки,

г) загальний час

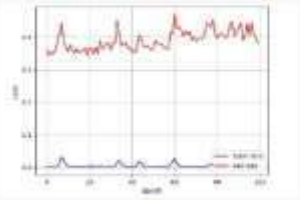
Особливості динаміки натискання клавіш

Гіперпараметр	Випробування									
	1	2	3	4	5	6	7	8	9	10
conv_1_filter	112	64	32	48	128	16	80	32	80	80
conv_1_kernel	2	2	2	3	2	2	2	2	2	3
conv_1_activation	R	R	R	S	R	S	R	T	R	T
conv_2_filter	64	112	32	32	48	64	64	96	112	128
conv_2_kernel	2	2	2	3	2	3	2	2	2	3
conv_2_activation	R	R	T	R	S	R	R	R	R	R
LSTM_units	112	32	112	96	128	112	96	80	64	32
dense_1_units	48	80	112	48	112	128	64	80	80	80
dense_activation	T	T	T	T	T	R	R	T	R	T
dropout_rate	0.1	0.1	0.3	0.1	0.3	0.1	0.1	0.4	0.3	0.3
Точність	0.9988	0.9969	0.9968	0.9955	0.9954	0.9937	0.9901	0.9898	0.9895	0.9893

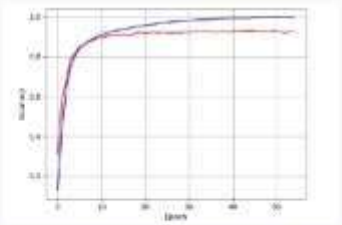
Раціональні гіперпараметри для максимальної точності



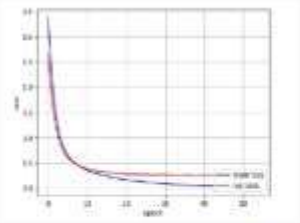
Залежність точності від кількості епох



Залежність втрат від кількості епох



Значення точності при ранній зупинці



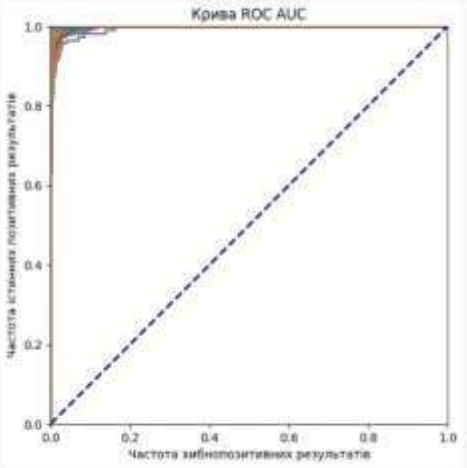
Значення втрат при ранній зупинці


```
[0.2559188902378082, 0.9307692050933838]
Accuracy = 0.9307692307692308
Precision = 0.9314594819701478
Recall = 0.9307692307692308
F1 Score = 0.9305327311472332
```

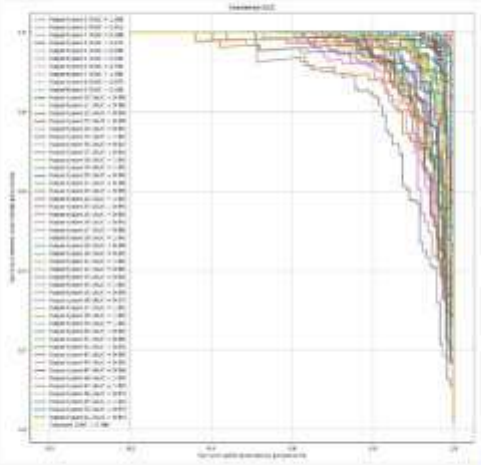
Розрахунок метрик моделі

	precision	recall	f1-score	support
0	1.00	0.99	1.00	117
1	0.84	0.85	0.85	119
2	0.91	0.96	0.93	113
3	0.93	0.92	0.92	125
4	0.97	0.97	0.97	116
5	0.91	0.83	0.87	139
6	0.94	0.84	0.88	141
7	0.98	0.97	0.97	132
8	0.93	0.91	0.92	128
9	0.92	0.93	0.92	145
...	...	...	...	...
44	0.89	0.91	0.90	121
45	0.90	0.90	0.90	122
46	0.98	0.98	0.98	122
47	0.96	0.98	0.97	122
48	0.90	0.97	0.93	119
49	0.97	0.98	0.97	117
50	0.98	0.90	0.94	123
51	0.94	0.81	0.87	111
accuracy			0.93	6240
macro avg	0.93	0.93	0.93	6240
weighted avg	0.93	0.93	0.93	6240

Звіт основних метрик класифікації

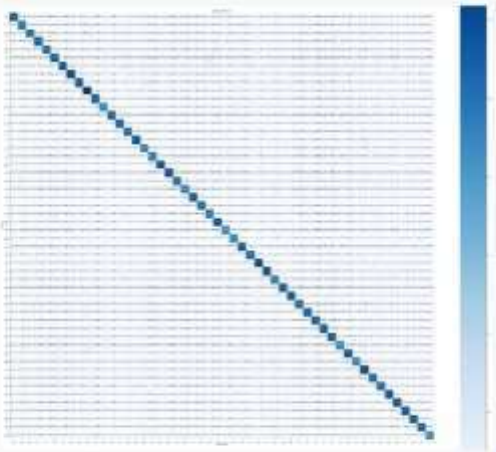


Графік ROC-кривої



Побудова AUC

Матриця плутанини для 12 користувачів



Матриця плутанини

ВИСНОВКИ

Дана робота присвячена розробці нейронної мережі для аутентифікації користувачів за клавіатурним почерком.

У цілому, дослідження показує, що нейромережева аутентифікація користувачів за клавіатурним почерком має великий потенціал у сфері захисту даних та ідентифікації користувачів. Запропонована модель може бути використана для підвищення рівня безпеки систем та збереження приватності користувачів в різних сферах. Такі дослідження сприяють розвитку технологій безпеки та можуть бути використані для майбутніх застосувань у різних галузях, де важлива надійна аутентифікація користувачів.

Список публікацій здобувача

Башков Є. О., Алтухова Т.В., Єжова Є. О. Розробка методу аутентифікації користувача на основі клавіатурного почерку / Є.О. Башков, Т.В. Алтухова, Є.О. Єжова // Наукові праці Донецького національного технічного університету. – Луцьк, 2023. - №2(35)'2022 - №1(36)'2023. – С. 61-68. DOI: 10.31474/1996-1588-2023-1-36-61-68

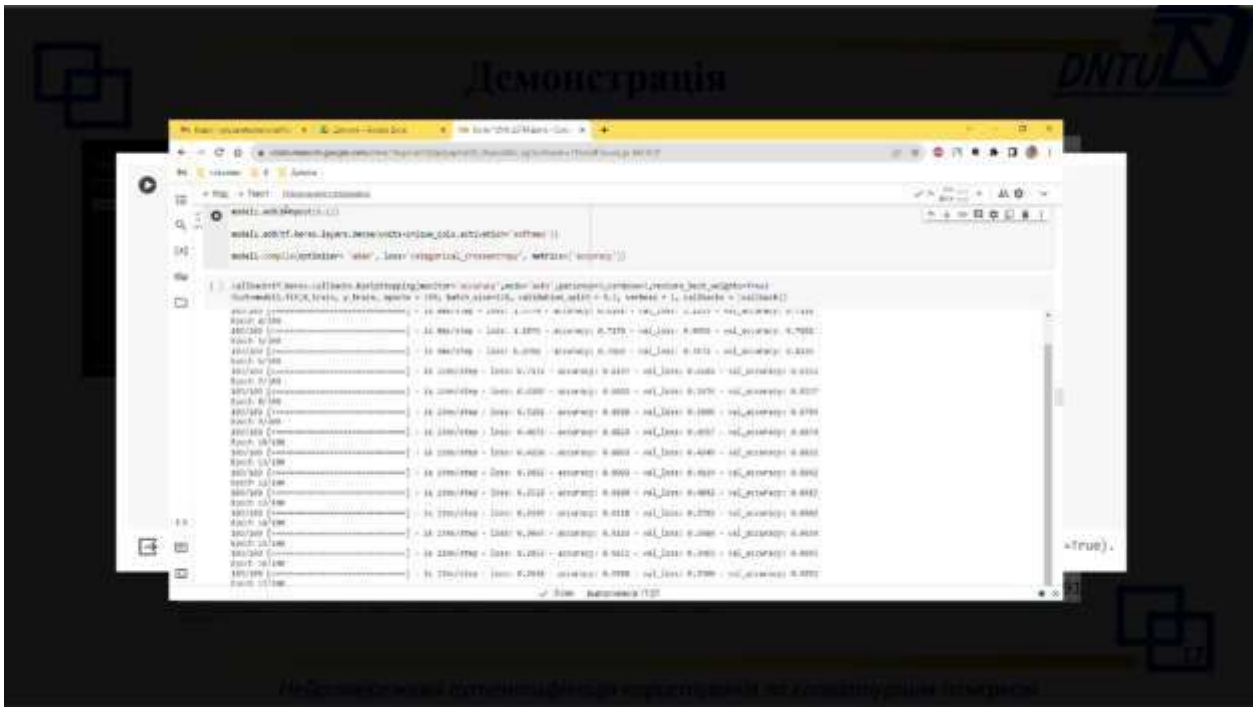
Єжова Є. О. Використання нейронних мереж для аутентифікації користувача за клавіатурним почерком / Є. О. Єжова // Моделювання і комп'ютерна графіка: зб. матер. Восьмої міжнар. наук.-техн. конф, 11-14 квітня 2023 р.: присв. 100-річчю з дня народж. Л. П. Фельдмана / Донецький національний технічний університет. – Луцьк – Київ: ДВНЗ «ДонНТУ», 2023. – С. 66-72

Єжова Є. О. Аналіз застосування сучасних методів розпізнавання облич з використанням нейронних мереж / Є. О. Єжова // «ТАК»: телекомунікації, автоматика, комп'ютерно-інтегровані технології: зб. доповідей Всеукр. наук.-практ. конф. молодих вчених, 1 грудня 2022 р. / ДВНЗ «ДонНТУ; відп. ред. Г.В. Ступак. – Луцьк: ДВНЗ «ДонНТУ», 2022. – С. 7-10

Єжова Є. О. Overview and problems of biometric authentication methods / Є. О. Єжова // Актуальні питання та перспективи проведення наукових досліджень: матеріали III Міжнародної студентської наукової конференції (Т. 3), м. Трускавець, 10 грудня, 2021 рік / ГО «Молодіжна наукова ліга». – Вінниця: ГО «Європейська наукова платформа», 2021. – С. 53–55 с. DOI 10.36074/liga-inter-10.12.2021

Список публікацій прийнятих до друку

1. Єжова Є. О. Нейронна мережа аутентифікації користувача за клавіатурним почерком / Є. О. Єжова // Електронні інформаційні ресурси: створення, використання, доступ: Зб. Матеріалів Міжнародної нак.-практ. Інтернет конф. (2023).


Демонстрація
DNTU

DNTU
Дякую за увагу!