Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Застосування згорткових нейронних мереж
для розпізнавання тексту і символів

Виконав:
студент 2 курсу, групи КІТм-20-1
Якимаха М. Є
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні
інтелектуальні технології

Керівник доц. Сердюк Н. М.

Допускається до захисту _____
(підпис)

Зав. кафедри _____ проф. Руденко О.Г.
(підпис)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____202_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Якимасі М. Є.

1. Тема роботи (проекту) Застосування згорткових нейронних мереж для розпізнавання тексту і символів
затверджена наказом університету від "_8_"____грудня_____2021р. № __1666Ст____
2. Термін подання студентом роботи до екзаменаційної комісії 15 грудня 2021р.
3. Вихідні дані до роботи (проекту) 1) Робоча машина: Ноутбук на базі процесора Intel Core i5 7200U з відео-картою Nvidia GeForce 940MX та 8 гігабайтами оперативної пам'яті
2) Операційна система: Windows 11 (64bit)
3) Середовище розробки згорткової нейронної мережі: PyCharm
4) Середовище розробки додатку для операційної системи Android: Android Studio

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз існуючих нейронних мереж та причини вибору згорткової нейронної мережі
2) Вибір середовища розробки згорткової мережі
3) Інсталяція та налаштування програмного забезпечення для розробки згорткової нейронної мережі
4) Розробка згорткової нейронної мережі
5) Аналіз середовищ, для розробки додатку для операційної системи Android
6) Інсталяція та налаштування програмного забезпечення для розробки додатку для Android
7) Розробка додатку
8) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри)
Демонстраційні матеріали. Слайди 1-13 ф. арк. А4

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача та узгодження теми проекту	08.11.2021	
2	Аналіз існуючих нейронних мереж та причини вибору згорткової нейронної мережі	09.11.2021	
3	Вибір середовища розробки нейронної мережі	10.11.2021	
4	Інсталяція та налаштування середовища розробки PyCharm	11.11.2021 – 13.11.2021	
5	Розробка згорткової нейронної мережі	13.11.2021 – 24.11.2021	
6	Аналіз середовищ розробки додатків для Android	25.11.2021	
7	Інсталяція та налаштування середовища розробки AndroidStudio	26.11.2021 – 28.11.2021	
8	Розробка додатку	28.11.2021 – 09.12.2021	
9	Перевірка виконаного проекту керівником	10.12.2020	
10	Захист проекту	16.12.2021	

Дата видачі завдання ___ _____20__ р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить: 13 аркушів презентації, 75 сторінок, 4 розділа, 26 рисунка, 1 додаток, 16 джерел посилань.

Темою кваліфікаційної роботи є застосування згорткових нейронних мереж для розпізнавання тексту і символів.

Метою кваліфікаційної роботи є розробка згорткової нейронної мережі, навчання моделі та розробка додатку для операційної системи Android при розпізнаванні рукописного вводу символу хірагани. Об’єктом дослідження є використання нейромережі для розпізнавання рукописного символу. Предметом дослідження є розпізнавання рукописного вводу складової японської азбуки хірагана.

У результаті виконання кваліфікаційної роботи було розглянуто складові згорткових нейронних мереж, розглянуто технології за допомогою яких можна реалізувати згорткову нейронну мережу. Навчена модель та розроблено додаток для перевірки працездатності навченої моделі у реальних умовах. Розроблено та відлагоджено код. Отримана модель для розпізнавання рукописних символів хірагани може бути використана у додатках з навчання японської мови, з можливістю навчання правопису символів, або з розпізнавання рукописних символів за допомогою камери чи зображень.

ПЕРСЕПТРОН, НЕЙРОННА МЕРЕЖА, НАВЧАННЯ, РОЗПІЗНАВАННЯ ТЕКСТУ, ДОДАТОК, TENSORFLOW, KERAS, PYTHON, ANDROID STUDIO, KOTLIN, AVD

ABSTRACT

The explanatory note of the qualification work contains: 13 sheets of presentations, 75 pages, 4 sections, 26 figures, 1 appendix, 16 sources of references.

The topic of the qualification work is the use of convolutional neural networks for text and character recognition.

The purpose of the qualification work is to develop a convolutional neural network, model learning and development of an application for the Android operating system for handwriting recognition of the hiragana symbol. The object of research is the use of neural networks to recognize handwritten symbols. The subject of the study is the recognition of handwritten input of the Japanese component of the alphabet.

As a result of the qualification work, the components of convolutional neural networks were considered, the technologies with which it is possible to implement a convolutional neural network were considered. Trained model and developed an application to test the performance of the trained model in real conditions. Developed and debugged code. The resulting model for recognizing handwritten characters hiragana can be used in applications for learning Japanese, with the ability to learn the spelling of characters, or for recognizing handwritten characters with a camera or images.

PERSEPTRON, NEURAL NETWORK, LEARNING, TEXT RECOGNITION, APPENDIX, TENSORFLOW, KERAS, PYTHON, ANDROID STUDIO, KOTLIN, AVD

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Комп'ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Застосування згорткових нейронних мереж
для розпізнавання тексту і символів

Виконав:
студент 2 курсу, групи КІТм-20-1
Якимаха М. Є.

Спеціальність 123 Комп'ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп'ютерні
інтелектуальні технології

Керівник доц. Сердюк Н. М.

2021 р.

АНОТАЦІЯ

У 2021 році нейронні мережі все частіше використовуються у повсякденному житті. Так, Google Translate може легко знаходити та розпізнавати текст на зображенні, а також переводити його, завдяки тому, що використовує технології нейронних мереж.

Актуальність цієї кваліфікаційної роботи має високий рівень через популяризацію азіатської культури, в особливості японської, тому, запотребуваність додатків з вивчання японської мови збільшується. Взагалі, таких додатків для навчання іноземних мов багато, але вузькоспеціалізованого, який було розроблено у цій кваліфікаційній роботі, практично немає.

Метою кваліфікаційної роботи є розробка згорткової нейронної мережі, навчання моделі та розробка додатку для операційної системи Android при розпізнаванні рукописного вводу символу хірагани.

Об’єктом дослідження є використання нейромережі для розпізнавання рукописного символу.

Предметом дослідження є розпізнавання рукописного вводу складової японської азбуки хірагана.

В якості методів дослідження було використано системний аналіз, статистичний метод дослідження результатів, метод навчання з учителем (на маркованих даних).

В процесі виконання кваліфікаційної роботи, було успішно розроблено згорткову нейронну мережу для навчання моделі, яка розпізнає рукописні символи японської складової азбуки хірагана. Модель була успішно навчена та має точність розпізнавання 99 відсотків, що є позитивним результатом. Цю модель можна використовувати у додатках, для навчання японської мови,

навчання правопису символів азбуки хірагана та розпізнавання рукописних символів хірагани на зображенні.

Для реалізації завдання кваліфікаційної роботи було досліджено предметну область нейронних мереж. Досліджено що таке персептрони, як вони працюють. Оглянуті проблеми навчання та їх вирішення, за допомоги сигмоїдних нейронів.

Розглянута архітектура нейронних мереж. Природним засобом проектування мережі являється кодування інтенсивностей пікселів зображення у вхідні нейрони. Якщо зображення має розмір 64x64, то маємо 4,069 = 64 * 64 вхідних нейрона. Вихідний шар має один нейрон, який містить вихідне значення, якщо воно більше, ніж 0.5, то на зображенні «9», інакше немає. У той час як проектування вхідних та вихідних шарів – досить просте завдання, вибрати архітектуру прихованих шарів вже складніше. Дослідники розробили безліч евристик проектування прихованих шарів, наприклад такі, які допомагають компенсувати кількість прихованих шарів проти часу навчання мережі.

Для швидкого навчання моделі була використана програмно-апаратна архітектура CUDA та бібліотеки Keras і Scikit-learn. CUDA (Compute Unified Device Architecture) – це архітектура паралельних обчислень, що значно збільшує обчислювальну продуктивність завдяки використанню графічних процесорів Nvidia.

CUDA SDK дозволяє розробникам реалізовувати на спеціальних спрощених діалектах мов програмування Ci, C++ та Fortran алгоритми, які можна здійснити на тензорних і графічних процесорах Nvidia. Архітектура CUDA дає можливість по своєму організовувати доступ до інструкцій графічного або тензорного прискорювача та керувати його пам'яттю. Функції, які прискорені за допомогою CUDA, можна викликати з різних мов розробки, зокрема у Python, MATLAB та інших.

Розробка програмного забезпечення для навчання моделі була здійснена на мові програмування Python. Модель навчалася та розроблялася у

середовищі розробки PyCharm. Ця мова програмування найбільш підходить для вирішення завдань штучного інтелекту. Додаток було розроблено на мові програмування Kotlin, у середовищі розробки Android Studio.

Навчання моделі було реалізовано за рахунок фреймворків TensorFlow, Keras та Scikit-learn. Перше було обрано за рахунок того, що це комплексна платформа з відкритим вихідним кодом для машинного навчання. Він має всеосяжну гнучку екосистему інструментів, бібліотек і ресурсів спільноти, яка дозволяє дослідникам просувати новітні досягнення в області машинного навчання, а розробникам легко створювати і розгортати додатки на основі машинного навчання. За допомоги TensorFlow, можна легко створювати і навчати моделі машинного навчання за допомогою інтуїтивно зрозумілих високорівневих API, таких як Keras, з активним виконанням, що забезпечує негайну ітерацію моделі і просту налагодження. Також є можливість легко навчати та розгортати моделі в хмарі, локально, в браузері або на пристрої, незалежно від того, яка мова використовується. Розроблений для швидкого експериментування з глибокими нейронними мережами, він фокусується на тому, щоб бути зручним, модульним та розширюваним.

Scikit-learn – один з найбільш широко використовуваних пакетів Python для Data Science і Machine Learning. Він дозволяє виконувати безліч операцій і надає безліч алгоритмів. Scikit-learn також пропонує відмінну документацію про своїх класи, методи і функції, а також опис використовуваних алгоритмів.

Для тестування моделі у реальних умовах використання додаток було розроблено для мобільної операційної системи Android у середовищі розробки Android Studio. Цей додаток включає в себе поле для графічного малювання символу, поле з результатом, у вигляді символу японського алфавіту хірагана та точністю визначення символу, а також кнопка Clear, яка дозволяє стерти вміст поля для малювання. Для реалізації додатка була обрана операційна система Android через те, що вона одна з найпоширеніших мобільних операційних систем. Для розробки додатків використовуються середовища розробки, такі як Eclipse, IntelliJ Idea та Android Studio. Було обране останнє

середовище через зручність розробки та його сумісність з мовою програмування Kotlin, та графічний інтерфейс, використовуваний для розташування об’єктів на сторінці додатку.

Для підтримки розробки програм в операційній системі Android, Android Studio використовує систему збірки на основі Gradle, емулятор, шаблони коду та інтеграцію з Github. Кожен проект в Android Studio має одну або кілька модальностей з вихідним кодом і файлами ресурсів. Ці модальності включають модулі програм Android, модулі бібліотеки та модулі Google App Engine.

Android Studio використовує функцію Instant Push для надсилання змін коду та ресурсів у запущену програму. Редактор коду допомагає розробнику писати код і пропонує його завершення, переломлення та аналіз. Програми, створені в Android Studio, потім компілюються у формат APK для надсилання в Google Play Store.

Основним критерієм вибору Android Studio є те, що ця середа розробки має функцію запуску емулятора операційної системи Android. Емулятор Android моделює пристрої Android на поточному комп’ютері, щоб була можливість тестувати свою програму на різних пристроях і рівнях API Android без необхідності мати кожен фізичний пристрій.

Емулятор надає майже всі можливості справжнього Android-пристрою. Є можливість моделювати вхідні телефонні дзвінки та текстові повідомлення, вказувати місце знаходження пристрою, моделювати різні швидкості мережі, імітувати обертання та інші апаратні датчики, доступ до Google Play Store та багато іншого. Емулятор поставляється з попередньо визначеними конфігураціями для різних телефонів, планшетів Android, Wear OS і пристроїв Android TV.

Тестування програми на емуляторі в деякому роді швидше та легше, ніж на фізичному пристрої. Наприклад, можна швидше передати дані на емулятор, ніж на пристрій, підключений через USB.

Для тестування навченої моделі у реальних умовах було розроблено додаток у середовищі розробки Android Studio, а також протестовано модель на правильність розпізнавання рукописних символів складової азбуки хірагана. Результати задовільні, розпізнавання працює без помилок.

Через те, що на даний момент популярність додатків з навчання іноземної, в особливості японської мови зростає, практичне використання навченої моделі може знайтись у сфері навчальних додатків підвищеного функціоналу, тобто з можливістю практикування правопису символів та літер.

ПЕРСЕПТРОН, НЕЙРОННА МЕРЕЖА, НАВЧАННЯ, РОЗПІЗНАВАННЯ ТЕКСТУ, ДОДАТОК, TENSORFLOW, KERAS, PYTHON, ANDROID STUDIO, KOTLIN, AVD

Публікації здобувача за темою роботи:

1. Якимаха М. Є. Особливості розпізнавання тексту та символів. *Радіоелектроніка та молодь у XXI столітті:* матеріали 25-го Міжнародного молодіжного форуму. Т. 5. – Харків: ХНУРЕ. 2021. – С. 177 – 178. URL: https://nure.ua/wp-content/uploads/2021/R&M/konferencija-5.pdf .

2. Якимаха М. Є. Використання нейромереж для розпізнавання тексту та символів. *Динаміка, рух та розвиток сучасної науки:* І Міжнародна студентська наукова конференція. Луцьк. 2021. С. 83 – 84. URL: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/05.03.2021/468

ЗМІСТ

ВСТУП

Станом на 2021 рік нейронні мережі все більше й більше зв’язуються із нашим життям. Так, Google Translate завдяки нейронній мережі може легко розпізнавати текст на зображенні та переводити його на іншу мову.

Актуальність цієї роботи полягає в тому, що зараз сильно популяризується азіатська культура, в особливості японська, тому, багато людей хочуть почати вивчати нові мови. Додатків, для навчання багато, але таких, яке було розроблено у цій кваліфікаційній роботі, практично немає.

Метою кваліфікаційної роботи є розробка згорткової нейронної мережі, навчання моделі та розробка додатку для операційної системи Android при розпізнаванні рукописного вводу символу хірагани.

Об’єктом дослідження є використання нейромережі для розпізнавання рукописного символу.

Предметом дослідження є розпізнавання рукописного вводу складової японської азбуки хірагана.

В якості методів дослідження було використано системний аналіз, статистичний метод дослідження результатів, метод навчання з учителем (на маркованих даних).

Інформація, яка була досліджена та використана для написання кваліфікаційної роботи була взята з електронних, наукових та бібліографічних джерел.

Структура роботи: Дослідження предметної області та актуальність проблеми; Розробка згорткової нейронної мережі та навчання моделі для розпізнавання рукописних символів японської складової азбуки хірагана; Огляд даних для розробки додатку на операційній системі Android, для тестування навченої моделі у реальних умовах; Розробка додатку для тестування натренованої моделі у реальних умовах.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АКТУАЛЬНІСТЬ ПРОБЛЕМИ

1.1 Теоретичні відомості щодо згорткових нейронних мереж

У 2021 році нейронні мережі використовуються у великій кількості сфер, в яких потрібні обчислення, пов'язані з розпізнаванням об'єктів на зображеннях, розпізнаванням рукописного вводу тексту та символів, передбаченням погоди. Найчастіше для цих задач використовуються нейронні мережі, які мають назву «згорткові нейронні мережі». Ці мережі складаються з персептронів.

Персептрон [1] був розроблений в 1950 р Френком Розенблатом. Персепрон приймає на вхід вектор $\overline{x} = \{x_1, x_2, x_3, \dots, x_N\}, x_i \in R$ і повертає деяке вихідне значення output $\in R$.

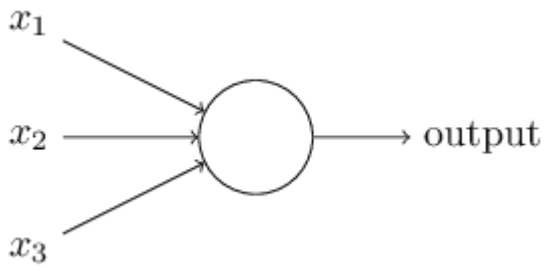

Рисунок 1.1 – Персептрон

Розенблат запропонував просте правило для обчислення вихідного значення. Він ввів поняття «значущості», далі «ваги» кожного вхідного значення $\overline{w} = \{w_1, w_2, w_3, \dots, w_N\}, w_i \in R$. В цьому випадку output буде залежати від того, чи буде $\sum_{i=1}^{N} x_i w_i$ більше або менше деякого порогового значення threshold $\in R$

$$output = \begin{cases} 0 if \sum_{i=1}^{N} x_i w_i \leq threshold \\ 1 if \sum_{i=1}^{N} x_i w_i > threshold \end{cases}$$
(1.1)

І це все, що потрібно. Варіюючи threshold і вектор ваг $\overline{w}$, можна отримати абсолютно різні моделі прийняття рішення.

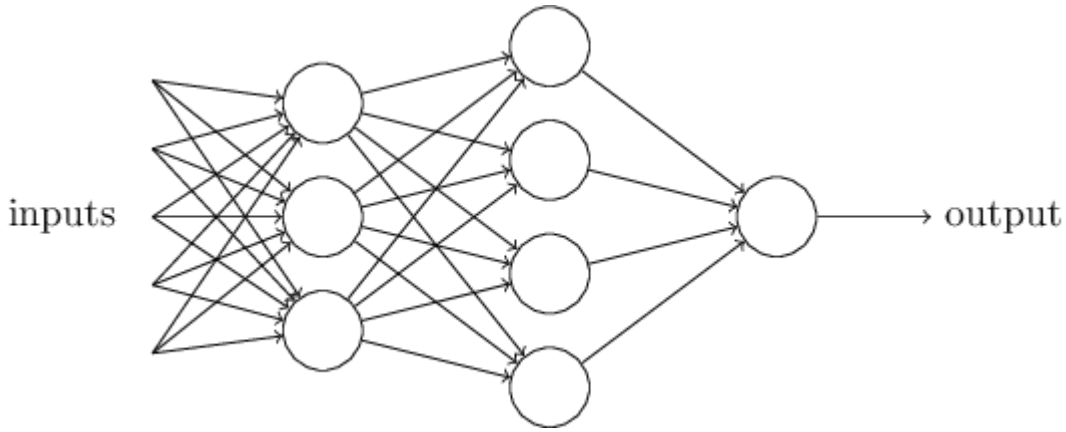

Рисунок 1.2 – Багатошарова нейронна мережа

Як видно на рисунку 1.2, мережа складається з декількох шарів нейронів [2, 3]. Перший шар називається вхідним шаром або рецепторами (Receptors, InputLayer), наступний шар – прихований (HiddenLayer), і останній – вихідний шар (OutputLayer). Умова $\sum_{i=1}^{N} x_i w_i > threshold$ досить громіздка, треба замінити $\sum_{i=1}^{N} x_i w_i$ на скалярний добуток векторів $\overline{x} \cdot \overline{w}$. Далі передбачимо b = threshold, назвавши його зміщенням персептрона або bias і перенесши b в ліву частину, вийде результат:

$$output = \begin{cases} 0 if \overline{x} \cdot \overline{w} + b \leq 0 \\ 1 if \overline{x} \cdot \overline{w} + b > 0 \end{cases}$$
(1.2)

Щоб дізнатися, як може працювати навчання, припустимо що була трохи змінена деяка вага або сталося зміщення в мережі. Потрібно, щоб ця невелика

зміна ваги викликала невелику відповідну зміну на виході з мережі. Схематично це виглядає так:

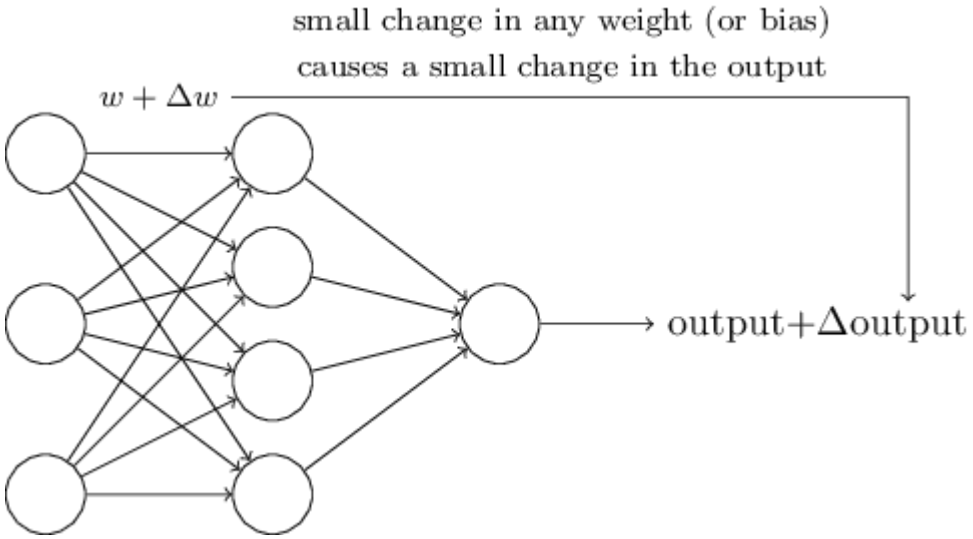

Рисунок 1.3 – Приклад зміни ваги

Якби це було можливо, то можна було б маніпулювати вагами в вигідну сторону і поступово навчати мережу, але проблема полягає в тому, що при деякій зміні вагів конкретного нейрона – його вихід може повністю «перевернутися» з 0 на 1. Це може привести до великої помилки прогнозу всієї мережі, але є спосіб обійти цю проблему, ввівши новий тип штучного нейрона, званий сігмоїдним нейроном. Сігмоїдні нейрони [4] подібні персептронам, але модифіковані так, що невеликі зміни в їх вагах і зміщення викликають лише невеликі зміни на їх виході. Структура сігмоїдного нейрона аналогічна, але тепер на вхід він може приймати $0 \leq x_i \leq 1, \forall x_i \in \overline{x}$ а на виході буде видавати $\sigma(\overline{x} \cdot \overline{w} + b)$, де:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.3}$$

персептрон і сигмоїдний нейрон мають багато спільного. Припустимо, що $z = \overline{x} \cdot \overline{w} + b \to \infty$, тоді $e^{-z} \to 0$ та з цього виходить, що $\sigma(z) \to 1$. Також вірне й

зворотне, якщо $z = \overline{x} \cdot \overline{w} + b \to -\infty$ то $e^{-z} \to \infty$ та $\sigma(z) \to 0$. Очевидно, що працюючи з сігмоїдним нейроном мається більш згладжений персептрон:

$$\Delta output \approx \sum_{i=1}^{N} \frac{\partial output}{\partial w_i} \Delta w_i + \frac{\partial output}{\partial b} \Delta b \quad (1.4)$$

1.2 Архітектура згорткових нейронних мереж

Проектування вхідних і вихідних шарів нейромережі – досить просте заняття. Для прикладу, відбувається визначення, зображена чи рукописна «9» на зображенні чи ні. Природним способом проектування мережі є кодування інтенсивностей пікселів зображення у вхідні нейрони. Якщо зображення має розмір 64x64, то маємо 4,069 = 64 * 64 вхідних нейрона. Вихідний шар має один нейрон, який містить вихідне значення, якщо воно більше, ніж 0.5, то на зображенні «9», інакше немає. У той час як проектування вхідних і вихідних шарів – досить просте завдання, вибір архітектури прихованих шарів – мистецтво. Дослідники розробили безліч евристик проектування прихованих шарів, наприклад такі, які допомагають компенсувати кількість прихованих шарів проти часу навчання мережі.

До сих пір, використовувалися нейронні мережі [5], в яких вихід з одного шару – використовувався як сигнал для наступного, такі мережі називаються прямими нейронними мережами або мережами прямого поширення (FeedForward). Однак існують і інші моделі нейронних мереж, в яких можливі петлі зворотного зв'язку. Ці моделі називаються рекурентними нейронними мережами (RecurrentNeuralNetwork). Рекурентні нейронні мережі були менш впливовими, ніж мережі з прямим зв'язком, почасти тому, що алгоритми навчання для рекурентних мереж (принаймні на сьогоднішній день) менш ефективні. Але рекурентні мережі як і раніше надзвичайно цікаві. Вони набагато ближче по духу до того, як працює мозок людини, ніж мережі з прямим зв'язком. І цілком можливо, що повторюючися мережі можуть вирішувати

важливі проблеми, які можуть бути вирішені з великими труднощами з допомогою мереж прямого доступу.

2 РОЗРОБКА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ ТА НАВЧАННЯ МОДЕЛІ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ ЯПОНСЬКОЇ АЗБУКИ ХІРАГАНА

2.1 Фреймворки та програмне забезпечення, використовувані для розробки згорткової нейронної мережі

TensorFlow – це комплексна платформа з відкритим вихідним кодом для машинного навчання [6]. Він має всеосяжну гнучку екосистему інструментів, бібліотек і ресурсів спільноти, яка дозволяє дослідникам просувати новітні досягнення в області машинного навчання, а розробникам легко створювати і розгортати додатки на основі машинного навчання. За допомоги TensorFlow, можна легко створювати і навчати моделі машинного навчання за допомогою інтуїтивно зрозумілих високорівневих API, таких як Keras, з активним виконанням, що забезпечує негайну ітерацію моделі і просту налагодження. Також є можливість легко навчати та розгортати моделі в хмарі, локально, в браузері або на пристрої, незалежно від того, яка мова використовується.

Keras – це бібліотека програмного забезпечення з відкритим кодом, яка написана на мові програмування Python. Keras виступає в якості надбудування для бібліотеки TensorFlow [7].

До версії 2.3 Keras підтримував декілька серверних систем, включаючи TensorFlow, Microsoft Cognitive Toolkit, Theano та PlaidML. Станом на версію 2.4 підтримується лише TensorFlow. Розроблений для швидкого експериментування з глибокими нейронними мережами, він фокусується на тому, щоб бути зручним, модульним та розширюваним. Він був розроблений в рамках дослідницьких робіт проекту ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), а його головним автором та супровідником є Франсуа Шолле, інженер Google. Шолле також є автором моделі глибокої нейронної мережі XCeption.

Scikit-learn – один з найбільш широко використовуваних пакетів Python для Data Science і Machine Learning. Він дозволяє виконувати безліч операцій і надає безліч алгоритмів. Scikit-learn також пропонує відмінну документацію про своїх класи, методи і функції, а також опис використовуваних алгоритмів.

Scikit-Learn підтримує:

- попередню обробку даних;
- зменшення розмірності;
- вибір моделі;
- регресії;
- класифікації;
- кластерний аналіз.

Він також надає кілька наборів даних, які можна використовувати для тестування ваших моделей.

Scikit-learn не реалізує все, що пов'язано з машинним навчанням. Наприклад, він не має комплексної підтримки для:

- нейронних мереж;
- самоорганізованих карт (мереж Кохонена);
- навчання асоціативним правилам;
- навчання з підкріпленням (reinforcement learning).

Scikit-learn заснований на NumPy і SciPy, тому необхідно зрозуміти хоча б ази цих двох бібліотек, щоб ефективно застосовувати Scikit-learn.

Scikit-learn - це пакет з відкритим вихідним кодом. Як і більшість матеріалів з екосистеми Python, він безкоштовний навіть для комерційного використання. Він ліцензований під ліцензією BSD.

OpenCV – бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом [8].

Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування

методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Бібліотека розроблена Intel і нині підтримується Willow Garage[en] та Itseez. Сирцевий код бібліотеки написаний мовою C++ і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua та інших. Може вільно використовуватися в академічних та комерційних цілях.

SciPy – це бібліотека Python з відкритим вихідним кодом, призначена для вирішення наукових та математичних проблем. Вона побудована на базі NumPy і дозволяє керувати данними, а також візуалізувати їх за допомогою різних високоуровневих команд.

PyCharm – це інтегроване середовище розробки для мови програмування Python [9]. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA. Це крос-платформне середовище розробки, сумісне з Windows, macOS та Linux.

Можливості цієї середи розробки:

- налагодження коду за допомогою PyDev;
- рефакторинг коду;
- нідтримка Git, SVN, Mercurial та інших систем контролю версіями;
- автодоповнення коду.

2.2 Розробка та навчання згорткової нейронної мережі

Для реалізації більш швидкого навчання згорткової мережі, було використано програмно-апаратну архітектуру CUDA та бібліотеки Keras та Scikit-learn. CUDA (Compute Unified Device Architecture) – це архітектура паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів фірми Nvidia [10].

CUDA SDK дозволяє програмістам реалізовувати на спеціальних спрощених діалектах мов програмування Ci, C ++ і Fortran алгоритми, здійснимі на графічних і тензорних процесорах Nvidia. Архітектура CUDA дає розробнику можливість на свій розсуд організовувати доступ до набору інструкцій графічного або тензорного прискорювача і управляти його пам'яттю. Функції, прискорені за допомогою CUDA, можна викликати з різних мов, в т.ч. Python, MATLAB та інших.

Tensorflow було використано для коректної роботи cudnn.

На представленому нижче коді, описується маркування даних:

```
Y_train = np.repeat(np.arange(num_classes), 160)

X_train, X_test, Y_train, Y_test = train_test_split(X_train, Y_train, test_size=0.2)

if K.image_data_format() == 'channels_first':
    X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
    X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
    X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

Перетворення класу векторів у бінарний клас матриць, тобто перетворення у дані, для тренування, які підтримуються keras-ом:

```
Y_train = np_utils.to_categorical(Y_train, num_classes)
Y_test = np_utils.to_categorical(Y_test, num_classes)

datagen = ImageDataGenerator(rotation_range=15, zoom_range=0.20)
datagen.fit(X_train)

model = Sequential()
```

Код моделі:

```
def mod():

    model.add(Conv2D(32,                                    kernel_size=(3,
3),kernel_initializer=my_init,input_shape=X_train.shape[1:]))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3),kernel_initializer=my_init))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))


    model.add(Conv2D(64, (3, 3),kernel_initializer=my_init))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3),kernel_initializer=my_init))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))


    model.add(Flatten())
    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes))
    model.add(Activation('softmax'))
```

Структура моделі:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
activation_1 (Activation)    (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 32)        9248
_________________________________________________________________
```

```
activation_2 (Activation)    (None, 60, 60, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 26, 26, 64)        36928
_________________________________________________________________
activation_4 (Activation)    (None, 26, 26, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 13, 13, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 10816)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2769152
_________________________________________________________________
activation_5 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 75)                19275
_________________________________________________________________
activation_6 (Activation)    (None, 75)                0
=================================================================
Total params: 2,853,419
Trainable params: 2,853,419
Non-trainable params: 0
```

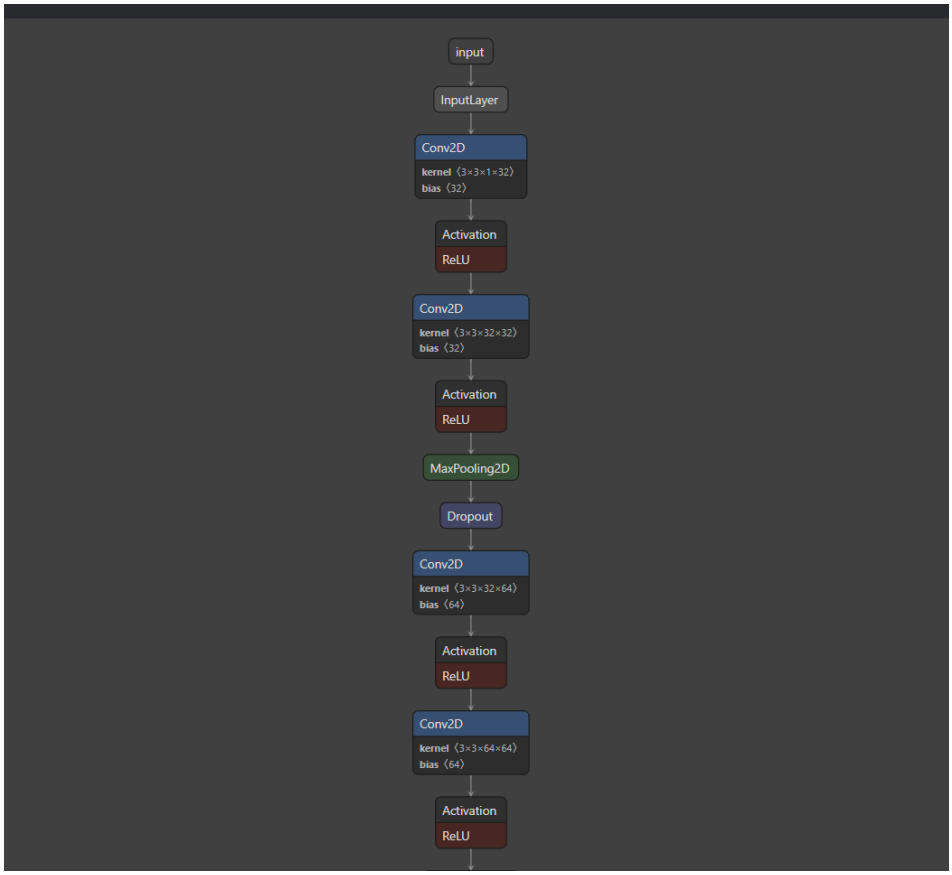

Рисунок 2.1 – Візуалізація структури навченої моделі (1 частина)

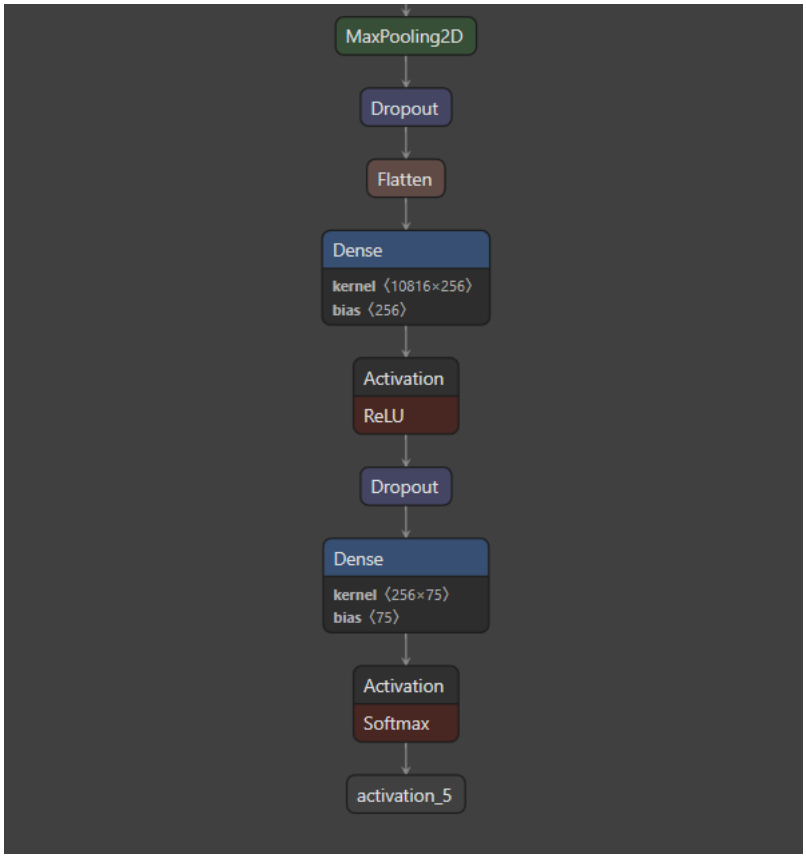

Рисунок 2.2 – Візуалізація структури навченої моделі (2 частина)

Збереження моделі:

```
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'hirahanaMod1.h5'
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)
```

Результати навчання моделі:

```
Test loss: 0.23362154187013706
Test accuracy: 0.9908333333333333
```

2.3 Перевірка коректної роботи навченої моделі для розпізнавання символів азбуки хірагана

Для перевірки роботи моделі, було розроблено наступний код програми:

Дані ієрогліфів алфавіту хірагана, для відображення розпізнаного символу:

```
kana=['あ','い','う','え','お','か','が','き','ゃ','ぎ','く','ぐ','け','げ
','こ','ご','さ','ざ','し','ゅ','じ','す','ず','せ','ぜ','そ','ぞ','た','だ
','ち','ょ','ぢ','つ','づ','て','で','と','ど','な','に','っ','ぬ','ね','の
','は','ば','ぱ','ひ','び','ぴ','ふ','ぶ','ぷ','へ','べ','ぺ','ほ','ぼ','ぽ
','ま','み','む','め','も','や','ゆ','よ','ら','り','る','れ','ろ','わ','を
','ん']
kana_num = len(kana)
print(kana_num)
labels = np.empty([0, kana_num], np.int)
```

Завантаження навченої моделі:

```
l_model = load_model('./saved_models/hirahanaMod1.h5')


ary = np.load("./dataset/etlgtobfutoji64.npz")['arr_0'].reshape([-1, 64, 64]).astype(np.float32) / 15
X_train = np.zeros([num_classes * 160, img_rows, img_cols], dtype=np.float32)
for i in range(num_classes * 160):

    X_train[i] = scipy.misc.imresize(ary[i], (img_rows, img_cols), mode='F')
plt.imshow(ary[160*1+150])
Y_train = np.repeat(np.arange(num_classes), 160)

X_train, X_test, Y_train, Y_test = train_test_split(X_train, Y_train, test_size=0.2)


if K.image_data_format() == 'channels_first':
    X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
    X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
    X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

Завантаження вхідного зображення та зміна його розширення на 64x64 пікселей:

```
images = np.empty([0, 64, 64], np.float32)
img_ori = Image.open('./sample_images/a.png')
```

```
resize_img = img_ori.resize((64, 64),Image.LANCZOS)
#img_gray = ImageOps.grayscale(resize_img)
img_gray=resize_img.convert("L")
img_b=img_gray.point(lambda x: 0 if x < 192 else x)
img_b=img_gray.point(lambda x: 255 if x >= 192 else x)
img_b.show()
img_ary = np.asarray(img_b)

img_ary = 255 - img_ary
img_ary[img_ary<110]=0
img_ary[img_ary>=110]=255
```

Результати розпізнавання:

Рисунок 2.3 – Вхідне зображення

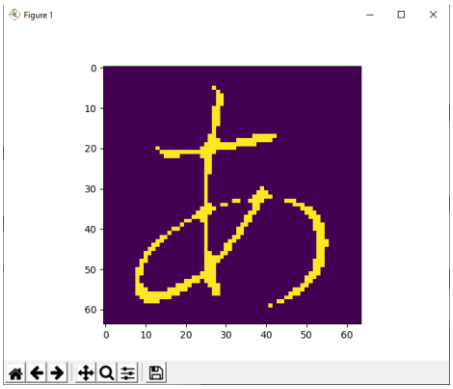

Рисунок 2.4 – Виділений контур

```
1 あ 97.34 %
```

2 お 1.40 %

3 ず 1.00 %

4 ゆ 0.26 %

Result: あ

3 ДОСЛІДЖЕННЯ ДАНИХ ДЛЯ РОЗРОБКИ ДОДАТКУ НА ОПЕРАЦІЙНІЙ СИСТЕМІ ANDROID

3.1 Android Studio та його можливості

Операційна система Android була обрана через те, що вона одна з найпоширеніших мобільних операційних систем. Для розробки додатків використовуються середовища розробки, такі як Eclipse, IntelliJ Idea та Android Studio. Було обране останнє середовище, через зручність розробки додатків, його сумісність з мовою програмування Kotlin та графічний інтерфейс, використовуваний для розташування об’єктів на сторінці додатку.

Android Studio – це офіційне інтегроване середовище розробки (IDE) для розробки додатків Android. Воно засновано на IntelliJ IDEA, інтегрованому середовищі розробки Java для програмного забезпечення, і містить інструменти для редагування коду та розробника.

Для підтримки розробки програм в операційній системі Android, Android Studio використовує систему збірки на основі Gradle, емулятор, шаблони коду та інтеграцію з Github. Кожен проект в Android Studio має одну або кілька модальностей з вихідним кодом і файлами ресурсів. Ці модальності включають модулі програм Android, модулі бібліотеки та модулі Google App Engine.

Android Studio використовує функцію Instant Push для надсилання змін коду та ресурсів у запущену програму. Редактор коду допомагає розробнику писати код і пропонує його завершення, переломлення та аналіз [11]. Програми, створені в Android Studio, потім компілюються у формат APK для надсилання в Google Play Store.

3.1.1 Можливості емуляції операційної системи Android в Android Studio

Android Studio має функцію запуску емулятора операційної системи Android. Емулятор Android [12] моделює пристрої Android на поточному комп'ютері, щоб була можливість тестувати свою програму на різних пристроях і рівнях API Android без необхідності мати кожен фізичний пристрій.

Емулятор надає майже всі можливості справжнього Android-пристрою. Є можливість моделювати вхідні телефонні дзвінки та текстові повідомлення, вказувати місцезнаходження пристрою, моделювати різні швидкості мережі, імітувати обертання та інші апаратні датчики, доступ до Google Play Store та багато іншого.

Тестування програми на емуляторі в деякому роді швидше та легше, ніж на фізичному пристрої. Наприклад, можна швидше передати дані на емулятор, ніж на пристрій, підключений через USB.

Емулятор поставляється з попередньо визначеними конфігураціями для різних телефонів, планшетів Android, Wear OS і пристроїв Android TV.

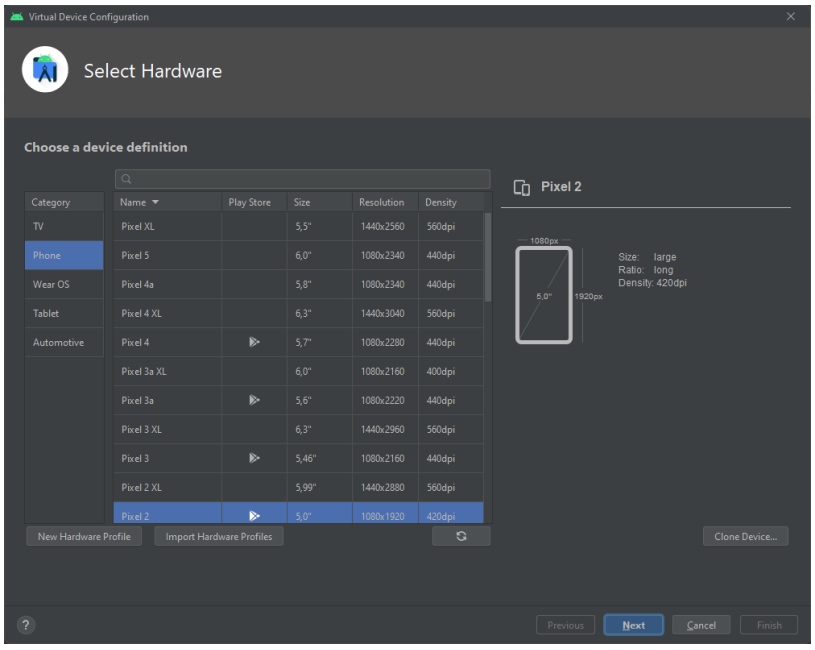

Рисунок 3.1 – Конфігурації для емуляції пристроїв

Емулятор можна використовувати вручну через його графічний інтерфейс користувача та програмно через командний рядок та консоль емулятора.

Рисунок 3.2 – Графічний інтерфейс користувача

Також емулятор має панель керування [13] з правого боку емулюючого пристроя. Кожна кнопка на панелі інструментів пов’язана з кнопками на клавіатурі, які можна визначити, навівши вказівник миші на кнопку та дочекавшись появи підказки, або за допомогою параметра довідки на розширеній панелі керування.

Панель керування емулятора має такі кнопки керування:

- Масштабування/Вихід;
- Ввімкнення/Вимикання;
- Підвищення гучності;
- Зниження гучності;
- Зміна положення пристрою;
- Знімок екрану;
- Режим приближення зображення;

– Кнопка «Додому»;
– Кнопка «Назад»;
– Кнопка «Мультизадачність»;
– Розширене керування.

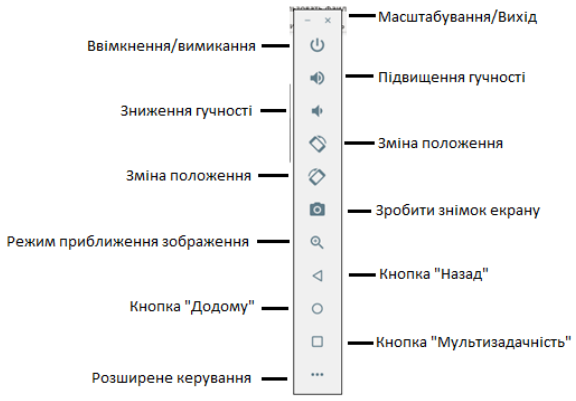

Рисунок 3.3 – Опції панелі керування

Масштабування/Вихід – сама верхня кнопка «х» на панелі інструментів виходить із сеансу емулятора, якщо натиснути, тоді як параметр «-» згортає все вікно.

Ввімкнення/Вимикання – кнопка живлення імітує апаратну кнопку живлення на фізичному пристрої Android. Натискання та відпускання цієї кнопки заблокує пристрій і вимкне екран. Натискання та утримування цієї кнопки ініціює послідовність запитів «Вимкнення живлення».

Підвищення/Зниження гучності – дві кнопки, які керують гучністю звуку відтворення в середовищі симулятора.

Зміна положення пристрою – повертає емульований пристрій між книжковою та альбомною орієнтаціями.

Знімок екрану – робить знімок вмісту екрана, який зараз відображається на екрані пристрою. Зроблене зображення зберігається в місці, зазначеному на екрані налаштувань розширеної панелі керування.

Режим приближення зображення – ця кнопка перемикає та виключає режим масштабування. Коли активний режим масштабування, кнопка панелі інструментів натискається, а вказівник миші відображається як збільшувальне скло, якщо навести курсор на екран пристрою. Клацання лівою кнопкою миші призведе до збільшення масштабу дисплея відносно вибраної точки на екрані, а повторне натискання збільшує рівень масштабування. І навпаки, клацання лівою кнопкою миші зменшує рівень масштабування. Вимкнення кнопки масштабування повертає дисплей до розміру за замовчуванням.

Клацання та перетягування в режимі масштабування визначить прямокутну область, до якої буде збільшено подання, коли буде відпущено кнопку миші.

У режимі масштабування видиму область екрана можна панорамувати за допомогою горизонтальних і вертикальних смуг прокрутки, розташованих у вікні емулятора.

Кнопка «Назад» – Імітує вибір стандартної кнопки «Назад» для Android. Як і з кнопками «Додому» та «Огляд», наведеними нижче, тих же результатів можна досягти, вибравши фактичні кнопки на екрані емулятора.

Кнопка «Додому» – імітує вибір стандартної кнопки Android «Додому».

Кнопка «Мультизадачність» – імітує вибір стандартної кнопки Android «Огляд», яка відображає поточні запущені програми на пристрої.

Розширене керування – відображає розширену панель керування, що дозволяє налаштувати такі параметри, як моделювання розташування та телефонної активності, потужність акумулятора, тип стільникової мережі та ідентифікація відбитків пальців.

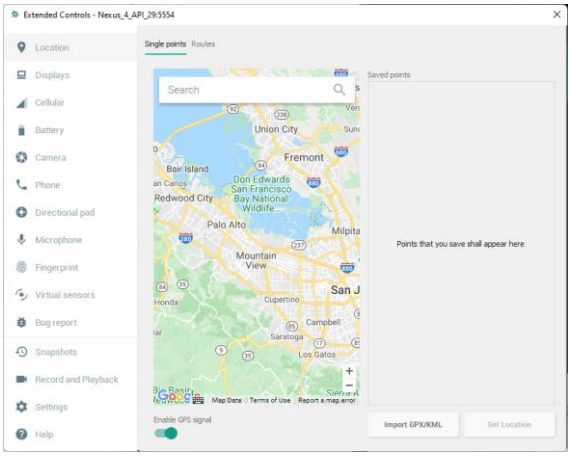

Рисунок 3.4 – Інтерфейс панелі розширеного керування

– Location;
– Cellular;
– Battery;
– Phone;
– Directional Pad;
– Fingerprint;
– Settings;
– Help.

Location – елемент керування розташуванням, який дозволяє надсилати на емулятор змодельовану інформацію про місцезнаходження у вигляді десяткових чи шістнадцятирічних координат. Інформація про місцезнаходження може мати форму окремого розташування або послідовності точок, що представляють рух пристрою, причому остання надається через файл у форматі GPS Exchange (GPX) або Keyhole Markup Language (KML).

При натисканні кнопки надіслання, до емулятора передається єдине розташування. Передача точок даних GPS починається після вибору кнопки «Відтворити», розташованої під таблицею даних. Швидкістю, з якою точки даних GPS надходять до емулятора, можна керувати за допомогою меню швидкості, розташованого поруч із кнопкою відтворення.

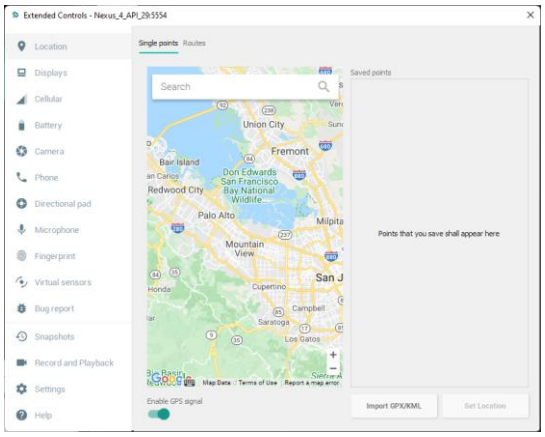

Рисунок 3.5 – Меню Location

Cellular – тип стільникового з’єднання, що моделюється, можна змінити на екрані налаштувань стільникового зв’язку. Доступні параметри для моделювання різних типів мереж (CSM, EDGE, HSDPA тощо) у доповнення до ряду сценаріїв голосу та даних, таких як роумінг та заборонений доступ.

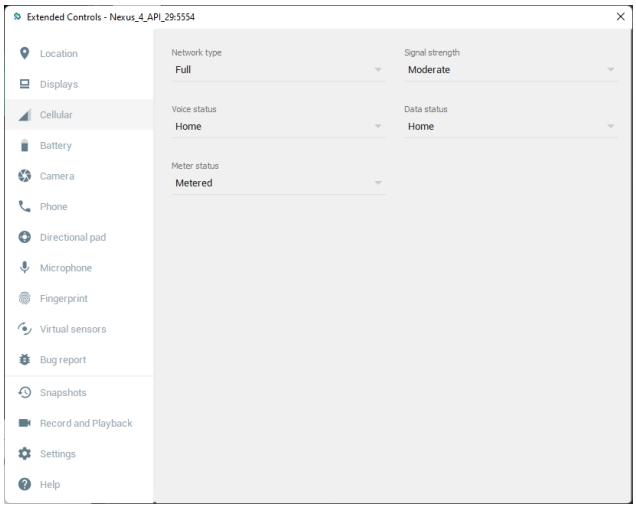

Рисунок 3.6 – Меню Cellular

Battery – на цій панелі розширеного екрана керування можна моделювати різноманітні стани батареї та умови зарядки, включаючи рівень заряду акумулятора, стан батареї та чи підключено зарядний пристрій змінного струму.

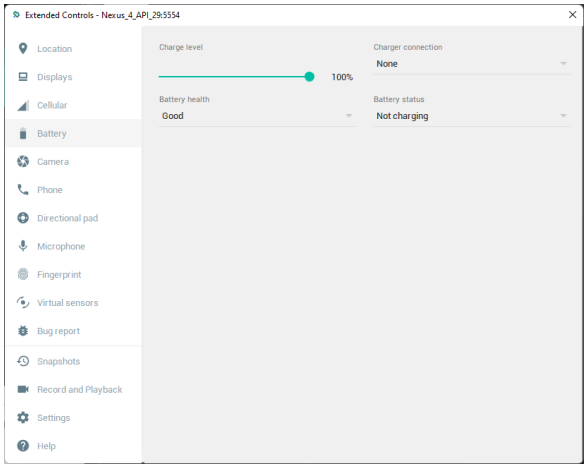

Рисунок 3.7 – Меню Battery

Phone – розширені елементи керування телефоном забезпечують дві дуже прості, але корисні симуляції в емуляторі. Перший варіант дозволяє імітувати вхідний дзвінок із зазначеного номера телефону. Це може бути особливо корисно під час тестування того, як програма обробляє переривання високого рівня такого характеру.

Другий варіант дозволяє моделювати отримання текстових повідомлень в рамках сеансу емулятора. Як і в реальному світі, ці повідомлення відображаються в програмі Message і запускають стандартні сповіщення в емуляторі.

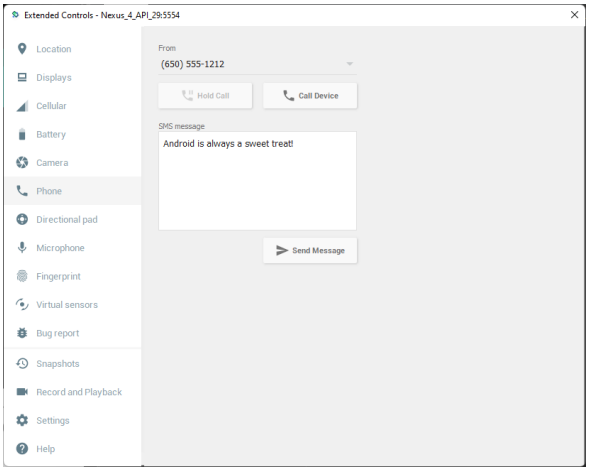

Рисунок 3.8 – Меню Phone

Directional Pad – панель керування (D-Pad) це додатковий набір елементів керування, вбудованих у пристрій Android або підключених іззовні

пристроїв керування, наприклад, ігровий контролер, який забезпечує керування напрямком ліворуч, праворуч, вгору і вниз.

Налаштування панелі напрямків дозволяють моделювати взаємодію D-Pad в емуляторі.

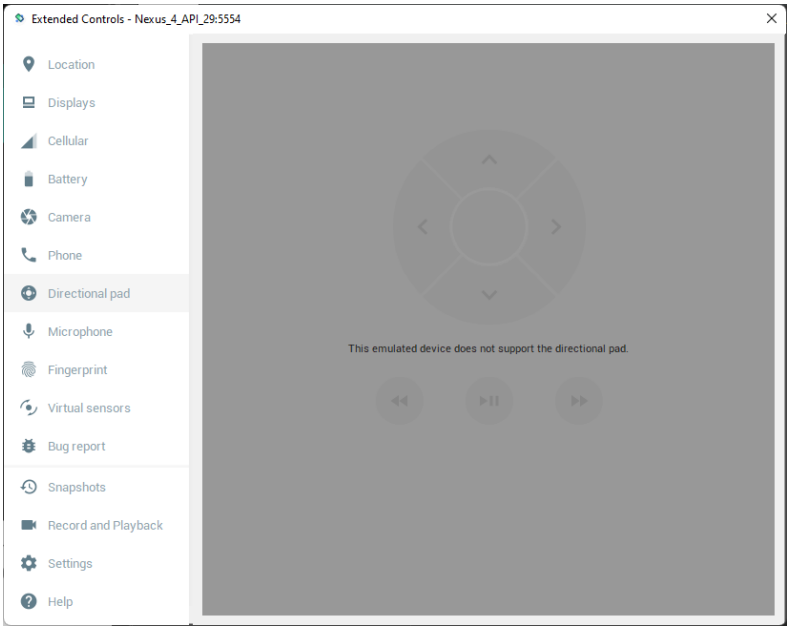

Рисунок 3.9 – Меню Directional Pad

Fingerprint – зараз багато пристроїв Android оснащені вбудованим обладнанням для виявлення відбитків пальців. З появою емулятора в Android Studio 2 тепер можна тестувати аутентифікацію відбитків пальців без необхідності тестувати програми на фізичному пристрої, що містить датчик відбитків пальців.

Емулятор дозволяє налаштувати до 10 змодельованих відбитків пальців і використовувати їх для перевірки автентифікації відбитків пальців у програмах Android. Щоб налаштувати імітовані відбитки пальців, треба почати із запуску емулятора, відкрити програму «Налаштування» та вибрати параметр «Безпека».

На екрані налаштувань безпеки вибрати параметр «Відбиток пальця». На отриманому інформаційному екрані натиснути кнопку «Продовжити», щоб перейти до екрана налаштування відбитків пальців. Перш ніж увімкнути захист

відбитків пальців, необхідно налаштувати резервний метод розблокування екрана (наприклад, PIN-код). Для цього треба натиснути кнопку «Налаштувати» блокування екрана, вибрати параметр PIN, ввести та підтвердити відповідний PIN-код і завершити процес введення PIN-коду.

Далі треба перейти до наступних екранів, доки додаток Налаштування не запитає відбитка пальця на сенсорі. На цьому етапі треба відкрити діалогове вікно розширеного керування, вибрати категорію «Відбиток пальця» на панелі ліворуч і переконатись, що на головній панелі налаштувань вибрано «Палець 1».

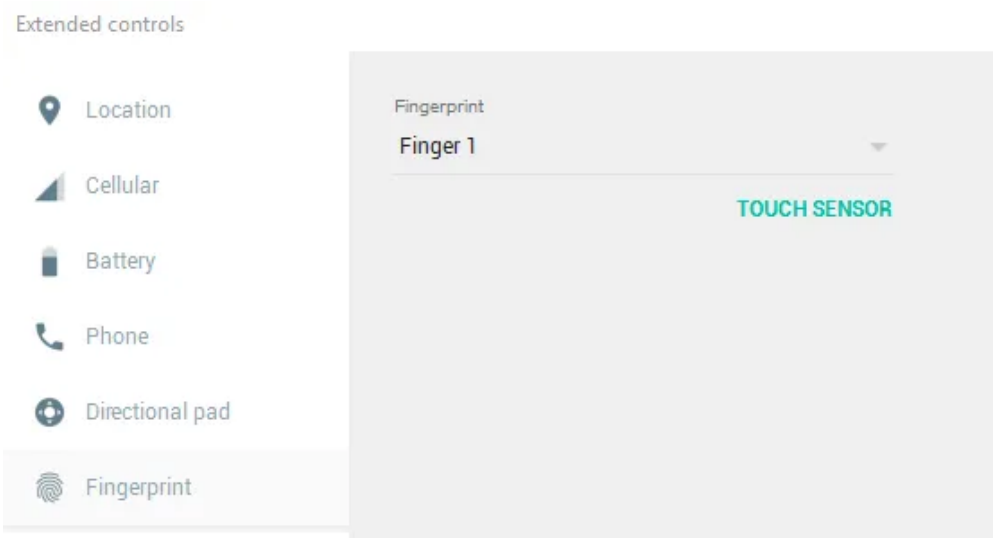

Рисунок 3.10 – Доданий відбиток пальця

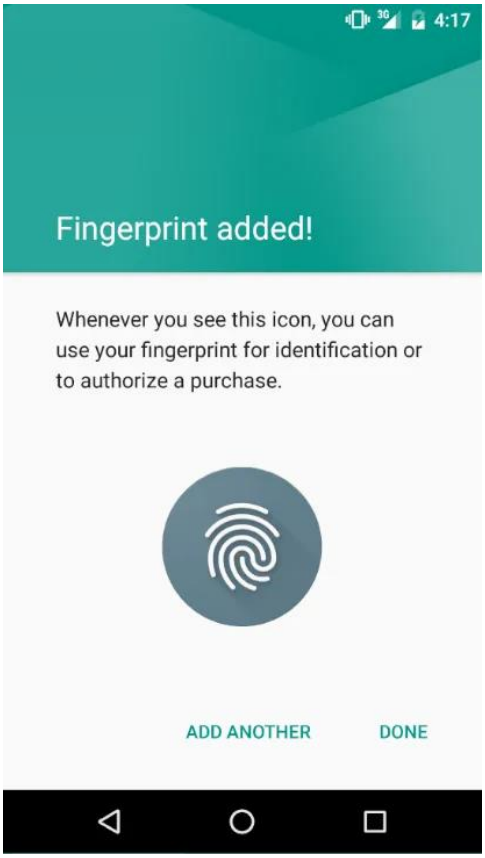

Рисунок 3.11 – Успішно доданий відбиток пальця

Settings – панель налаштувань містить невелику групу параметрів конфігурації. цю панель можна використовувати, щоб вибрати темнішу тему для панелі інструментів і розширеної панелі керування, вказати розташування у файловій системі, куди потрібно зберігати знімки екрана, і вибрати інший шлях SDK, який буде використовуватися під час запуску програм в емуляторі.

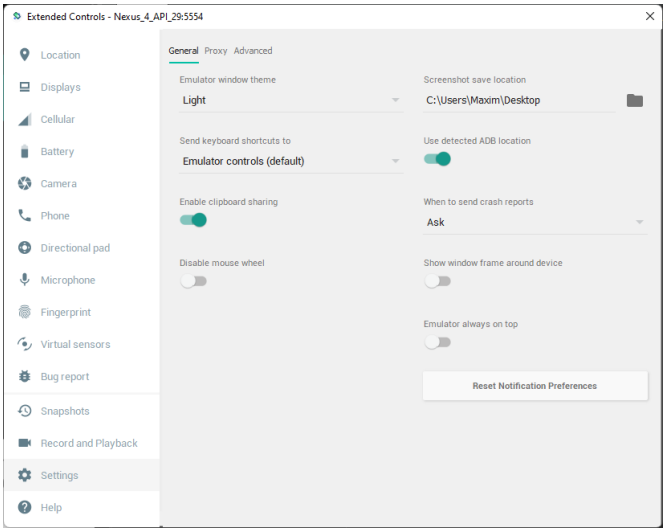

Рисунок 3.12 – Settings

Help – екран довідки містить три підпанелі, що містять список комбінацій клавіш, посилання для доступу до онлайн-документації емулятора, помилки у файлі та надсилання відгуків, а також інформацію про версію емулятора.

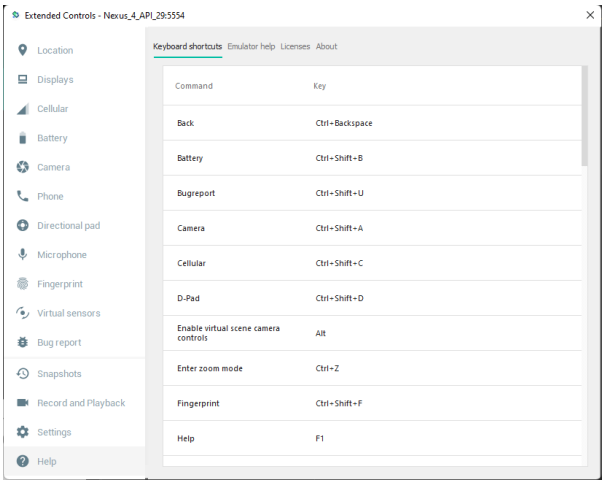

Рисунок 3.14 – Меню Help

Щоб підвищити продуктивність емулятора, Google додав можливість для образу системи Android, що працює в емуляторі, використовувати кілька ядер у процесорі комп’ютерної системи, на якій він працює. Щоб вказати кількість ядер, використовуваних емулятором, треба вимкнути емулятор, завантажити

AVD Manager і відредагувати конфігурацію емулятора, натиснувши відповідний значок олівця в списку емуляторів. На екрані «Конфігурація віртуального пристрою» треба натиснути кнопку «Показати додаткові параметри» та знайти меню «Багатоядерний ЦП» у розділі «Емульована продуктивність» на панелі. У цьому меню потрібно вибрати кількість ядер, які будуть використовуватися емулятором. Загальна кількість доступних ядер буде залежати від архітектури ЦП в системі.

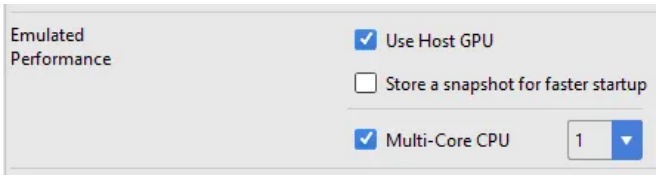

Рисунок 3.15 – Інтерфейс вибору кількості ядер

3.1.2 Віртуальні Android пристрої

Кожен екземпляр емулятора Android використовує віртуальний пристрій Android (AVD) [14] для визначення версії Android та характеристик апаратного забезпечення модельованого пристрою. Щоб ефективно протестувати свою програму, слід створити AVD, який моделює кожен пристрій, для якого призначена ваша програма. Для створення AVD та керування ними скористайтеся диспетчером AVD.

Кожен AVD функціонує як незалежний пристрій із власним приватним сховищем для даних користувача, SD-карти тощо. За замовчуванням емулятор зберігає дані користувача, дані SD-карти та кеш-пам'ять у каталозі, характерному для цього AVD. Коли запускається емулятор, він завантажує дані користувача та дані SD-карти з каталогу AVD.

Також, емулятор має можливість швидкого завантаження, називаємого як «Quick Boot». Ця функція дозволяє відновити сеанс емулятора Android менш ніж за 6 секунд. Коли вперше запускається віртуальний пристрій Android (AVD)

за допомогою емулятора Android, він повинен виконати холодне завантаження (як увімкнення пристрою), але наступні запуски відбуваються швидко, і система відновлюється до стану, в якому був закритий емулятор останній раз (подібно до пробудження пристрою). Це було досягнено, завдяки повній переробці старої архітектури знімків емулятора для роботи з віртуальними датчиками та прискоренням графічного процесора. Додаткове налаштування не потрібно, оскільки швидке завантаження ввімкнено за замовчуванням.

Це середовище розробки було обрано через те, що воно розповсюджується абсолютно безоплатно, а також має дуже велику кількість користувачів, що, в свою чергу, впливає на кількість навчального матеріалу.

3.2 Мова програмування Kotlin

Kotlin – це мова програмування з кросплатформним та статично типізованим виходом [15]. Вона спеціально розроблена для взаємодії з мовою Java. Ця мова в основному націлена на JVM, тобто віртуальну машину Java, а також підтримує JavaScript і власний код. Kotlin Language розроблений JetBrains у 2011 році. Google офіційно підтримує мову kotlin для розробки мобільних додатків спеціально для додатків Android.

Вона також має підтримку від Android Studio. Найкращі розробники додатків для Android прийняли мову Kotlin через його просту сумісність з Java. Звичайною мовою можна сказати, що ця мова допомагає розробникам не починати свою роботу з нуля, вони можуть легко поєднувати свою роботу з кодами Java за допомогою цієї мови. Kotlin – це не тільки мова розробки для Android, але також використовується для настільних програм та серверів.

Java – це найстаріша мова програмування, яка за останні 22 роки пропрацювала в області додатків. За цей довгий час Java надала розробникам безліч функцій для додатків Android. Згодом у цій старій мові з'являються деякі недоліки, тому з цього моменту Kotlin виходить на ринок і очолює всі інші мови зі своїми перевагами та функціями, а саме:

– Легкий процес навчання;
– Повна сумісність з мовою програмування Java;
– Виняток нульового покажчика;
– Функціональність швидкого запуска додатків;
– Безоплатна модель розповсюдження.

Провідним розробникам додатків для Android також потрібно багато часу, щоб вивчити будь-яку нову мову. Кожна мова має свій власний робочий процес та методи, які відрізняються від інших мов, тому вивчення нової мови займає дуже багато часу, але у випадку з Kotlin це зовсім інше: його легко зрозуміти, легко вивчити, зручна для користувача мова.

Основна ідея запуску цієї мови – зробити її сумісною з Java. Це просто означає, що провідні розробники програм для Android можуть використовувати ряд бібліотек Java для написання коду мови Kotlin, а також легко згенерувати код Java з цієї мови Kotlin за допомогою конвертерів.

Ця функція Kotlin допомагає розробникам зберігати знання про кодування Java, і при звичайному навчанні компанії з розробки програм для Android можуть використовувати своїх старих розробників для того ж, їм не потрібна ціла нова команда для цієї нової мови.

Через те, що Java достатньо стара, в ній є деякі обмеження, одна з найбільших проблем, з якими стикаються розробники програм при використанні Java, – це виняток із нульовою точкою. Через цю проблему мобільні додатки стикаються з серйозною проблемою в середині операції, але Kotlin прагне усунути цю проблему та запобігти збоям додатків Android у середині операції. Ця особливість робить мову Kotlin кращою за інші мови програмування.

Kotlin – це мова мобільних програм з відкритим вихідним кодом, тому немає необхідності витрачати коштовні гроші на цю мову програмування. Найпростіший спосіб використовувати цю мову додатків високого класу – перетворити її за допомогою інструмента перетворення Java на Kotlin, який може перетворити існуючі файли Java на прості коди Kotlin. Перетворення

складних кодів у простішу форму не потребує часу та безпечно. Також, Kotlin – це нова мова додатків сучасної епохи, і в міру її оновлення вона має багато нових функцій, які відрізняються і хороші від старих мов, таких як Java. Він має підтримку програмного забезпечення Android Studio, і тепер це офіційна мова розробки мобільних додатків Google. Завдяки оновленим функціям, Kotlin на крок попереду в розробці додатків для Android. Завдяки цим унікальним функціям простого методу навчання, швидкої роботи, сумісності з Java, безпеки та бездоганності, меншої вартості та багато іншого, він перевершує всі інші мови. Всі компанії, що займаються розробкою програм для Android, тепер переходять на мову Kotlin. Таким чином, можна сказати, що Kotlin – найкращий вибір для розробки індивідуальних мобільних додатків.

Kotlin клас компактний в порівнянні з іншими мовами, включаючи Java. Через компактність класу обсяг коду, який потрібно написати мовою Kotlin, також менший, ніж іншими мовами. І звістно, що менший код вимагає менше часу для запуску, і ця функція робить його швидше, ніж інші. Це мова з низьким рівнем кодування, тому вартість розробки програм також невисока. Ця функція робить його альтернативою Java у сучасну епоху.

4 РОЗРОБКА ДОДАТКУ ДЛЯ ТЕСТУВАННЯ НАТРЕНОВАНОЇ МОДЕЛІ У РЕАЛЬНИХ УМОВАХ

4.1 Інтерфейс розробленого додатку

Було розроблено простий додаток, для тестування натренованої моделі, для розпізнавання рукописного символу складової азбуки хірагана. Цей додаток включає в себе поле для графічного малювання символу, поле, з результатом, у вигляді символу японського алфавіту хірагана, та точність визначення символу, а також кнопка «Clear», яка дозволяє стерти вміст поля для малювання.

Рисунок 4.1 – Інтерфейс додатку

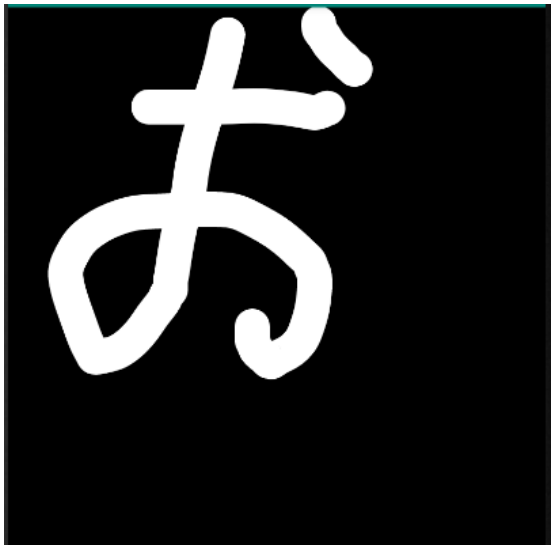

Рисунок 4.2 – Поле для малювання символу

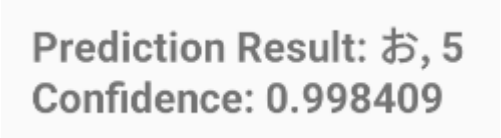

Рисунок 4.3 – Результат визначення

4.2 Бібліотеки, використовувані в додатку

Для роботи додатку з розробленою нейронною мережею, було написано ряд функцій, та використано такі бібліотеки [16]:

- android.content.Context;
- android.content.res.AssetManager;
- android.graphics.Bitmap;
- android.util.Log;
- com.google.android.gms.tasks.Task;
- com.google.android.gms.tasks.TaskCompletionSource;
- java.io.FileInputStream;
- java.io.IOException;
- java.nio.ByteBuffer;

- java.nio.ByteOrder;
- java.nio.channels.FileChannel;
- java.util.concurrent.ExecutorService;
- java.util.concurrent.Executors;
- org.tensorflow.lite.Interpreter.

android.content.Context – інтерфейс забеспечуючий доступ до глобальної інформації про середовище програми. Це абстрактний клас, реалізація якого забезпечується системою Android. Він надає доступ до ресурсів і класів, що стосуються програми, а також виклики для операцій на рівні програми, таких як запуск, трансляція та отримання намірів тощо.

android.content.res.AssetManager – надає доступ до необроблених файлів активів програми. Цей клас представляє API нижнього рівня, який дозволяє відкривати та читати необроблені файли, які були в комплекті з програмою, як простий потік байтів.

android.graphics.Bitmap – клас, який надає доступ до методів роботи з растровим зображенням.

android.util.log – API для відправки результатів журналу. Як правило, для запису журналів слід використовувати методи Log.v(), Log.d(), Log.i(), Log.w() і Log.e(). Потім можна переглянути журнали в logcat.

com.google.android.gms.tasks.TaskCompletionSource – надає можливість створювати API на основі завдань. Треба використовувати TaskCompletionSource, щоб встановити результат або виняток для завдання, поверненого з асинхронного API.

java.io.FileInputStream – FileInputStream отримує вхідні байти з файлу у файловій системі. Доступні файли залежать від середовища власника. FileInputStream призначений для читання потоків необроблених байтів, таких як дані зображення.

java.io.ioexception – це виключення введення/виводу, і вони виникають щоразу, коли операція введення або виведення не виконується або

інтерпретується. Наприклад, якщо хтось намагається прочитати щось у неіснуючому файлі, Java викличе виключення вводу-виводу.

java.nio.ByteBuffer – Java IO з використанням потоково-орієнтованих API виконується за допомогою буфера як тимчасового зберігання даних у просторі користувача. Дані, зчитовані з диска DMA, спочатку копіюються в буфери в просторі ядра, які потім передаються в буфер у просторі користувача. Тому є накладні витрати. Уникаючи його, можна досягти значного підвищення продуктивності. Можна було б пропустити цей тимчасовий буфер у просторі користувача, якби був спосіб прямого доступу до буфера в просторі ядра. Java NIO пропонує спосіб зробити це.

ByteBuffer є одним із кількох буферів, що надаються Java NIO. Це просто контейнер або резервуар для зчитування або запису даних. Наведена вище поведінка досягається шляхом виділення прямого буфера за допомогою API allocateDirect() на буфері.

java.nio.ByteOrder – цей клас визначає методи для читання та запису значень усіх інших примітивних типів, крім булевих. Примітивні значення транслюються в або з послідовності байтів відповідно до поточного порядку байтів буфера, який можна отримати та змінити за допомогою методів порядку. Конкретні порядки байтів представлені екземплярами класу ByteOrder. Початковий порядок буфера байтів завжди BIG_ENDIAN.

java.nio.channels.FileChannel – Канал для читання, запису, відображення та маніпулювання файлом. Файловий канал – це SeekableByteChannel, який підключений до файлу. Він має поточну позицію у своєму файлі, яку можна як запитувати, так і змінювати. Сам файл містить послідовність байтів змінної довжини, які можна читати та записувати, і поточний розмір яких можна запитати. Розмір файлу збільшується, коли байти записуються понад його поточний розмір; розмір файлу зменшується, коли його обрізають. Файл також може мати деякі пов’язані метадані, такі як дозволи доступу, тип вмісту та час останньої зміни; цей клас не визначає методи доступу до метаданих.

java.util.concurrent.ExecutorService – У багатопоточний пакет concurrent для управління потоками включено засіб, що називається сервісом виконання ExecutorService. Даний засіб служить альтернативою класу Thread, призначеному для управління потоками. На основі сервісу виконання інтерфейсу Executor, в якому визначено один метод. При виклику методу execute виконується поток thread. Тобто, метод execute запускає заданий потік на виконання.

java.util.concurrent.Executors – Фабричні та допоміжні методи для класів Executor, ExecutorService, ScheduledExecutorService, ThreadFactory та Callable, визначені в цьому пакеті. Цей клас підтримує такі види методів:

- Методи, які створюють і повертають ExecutorService, налаштовані із загальнокорисними параметрами конфігурації;
- Методи, які створюють і повертають ScheduledExecutorService, налаштовані із загальнокорисними параметрами конфігурації;
- Методи, які створюють і повертають "загорнутий" ExecutorService, який вимикає реконфігурацію, роблячи недоступними спеціальні для реалізації методи;
- Методи, які створюють і повертають ThreadFactory, який встановлює новостворені потоки у відомий стан;
- Методи, які створюють і повертають Callable з інших форм, подібних до закриття, тому їх можна використовувати в методах виконання, які вимагають Callable.

org.tensorflow.lite.Interpreter – Класовий драйвер для виведення моделей за допомогою TensorFlow Lite. Інтерпретатор інкапсулює попередньо навчальну модель TensorFlow Lite, яка виконує операції для виведення моделі. Наприклад, якщо модель приймає тільки один ввод і повертає тільки один висновок.

4.3 Розроблений клас на мові програмування «Kotlin», для зчитування вхідних форм з навченої моделі

У процесі роботи, був розроблений клас DigitClassifier, він містить такі функції, як initialize, яка додає інтерпретатор TensorFlow Lite як поле, завантаження моделі TensorFlow Lite та ініціалізація інтерпретатору, зчитування вхідних форм з файлу навченої моделі.

На представленому нижче коді визначена функція ініціалізації:

```
private var interpreter: Interpreter? = null
  var isInitialized = false
    private set

  private val executorService: ExecutorService =
Executors.newCachedThreadPool()

  private var inputImageWidth: Int = 0
  private var inputImageHeight: Int = 0
  private var modelInputSize: Int = 0

  fun initialize(): Task<Void> {
    val task = TaskCompletionSource<Void>()
    executorService.execute {
      try {
        initializeInterpreter()
        task.setResult(null)
      } catch (e: IOException) {
        task.setException(e)
      }
    }
    return task.task
  }
```

На представленому нижче коді визначена функція ініціалізації інтерпретатору:

```
private fun initializeInterpreter() {

val assetManager = context.assets
val model = loadModelFile(assetManager, "hiragana.tflite")
val interpreter = Interpreter(model)

val inputShape = interpreter.getInputTensor(0).shape()
inputImageWidth = inputShape[1]
inputImageHeight = inputShape[2]
modelInputSize = FLOAT_TYPE_SIZE * inputImageWidth *
inputImageHeight * PIXEL_SIZE

this.interpreter = interpreter

isInitialized = true
Log.d(TAG, "Initialized TFLite interpreter.")
}
```

На представленому нижче коді визначена функція підключення файлу навченої моделі:

```
private fun loadModelFile(assetManager: AssetManager, filename: String):
ByteBuffer {
    val fileDescriptor = assetManager.openFd(filename)
    val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
    val fileChannel = inputStream.channel
    val startOffset = fileDescriptor.startOffset
    val declaredLength = fileDescriptor.declaredLength
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
declaredLength)
  }
```

На представленому нижче коді визначена функція визначення намальованого в додатку символа:

```
    private fun classify(bitmap: Bitmap): String {
```

```
        check(isInitialized) { "TF Lite Interpreter is not initialized
yet." }
    input shape.
        val resizedImage = Bitmap.createScaledBitmap(
          bitmap,
          inputImageWidth,
          inputImageHeight,
          true
        )
        val byteBuffer = convertBitmapToByteBuffer(resizedImage)

        val output = Array(1) { FloatArray(OUTPUT_CLASSES_COUNT) }

        interpreter?.run(byteBuffer, output)

        val result = output[0]
       val maxIndex = result.indices.maxByOrNull { result[it] } ?: -1
        val resultString =
          "Prediction Result: %s, %d\nConfidence: %2f"
            .format(label[maxIndex], maxIndex + 1, result[maxIndex])

        return resultString
      }
```

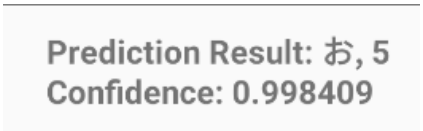

Рисунок 4.4 – Результат роботи функції визначення намальованого символу

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи, було успішно розроблено згорткову нейронну мережу для навчання моделі, яка розпізнає рукописні символи японської складової азбуки хірагана. Модель була успішно навчена та має точність розпізнавання 99 відсотків, що є позитивним результатом.

Для тестування навченої моделі у реальних умовах було розроблено додаток у середовищі розробки Android Studio, а також протестовано модель на правильність розпізнавання рукописних символів складової азбуки хірагана.

Результати задовільні, розпізнавання працює без помилок.

Через те, що на даний момент, популярність додатків з навчання іноземної, в особливості японської мови зростає, практичне використання навченої моделі може знайтись у сфері навчальних додатків підвищеного функціоналу, тобто з можливістю практикування правопису символів та літер.

ПЕРЕЛІК ПОСИЛАНЬ

1. Минский, М., Пейперт С. Перцептроны: книга. Москва: Мир, 1971. 261 с.
2. Якимаха М. Є. Особливості розпізнавання тексту та символів. *Радіоелектроніка та молодь у XXI столітті:* матеріали 25-го Міжнародного молодіжного форуму. Т. 5. – Харків: ХНУРЕ. 2021. – С. 177 – 178. URL: https://nure.ua/wp-content/uploads/2021/R&M/konferencija-5.pdf .
3. Якимаха М. Є. Використання нейромереж для розпізнавання тексту та символів. *Динаміка, рух та розвиток сучасної науки:* І Міжнародна студентська наукова конференція. Луцьк. 2021. С. 83 – 84. URL: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/05.03.2021/468 .
4. Каллан Р. Основные концепции нейронных сетей: книга. Москва: Вильямс, 2001. 288 с.
5. Беркинблит М. Б. Нейронные сети: книга. Москва: МИРОС та ВЗМШ РАО, 1993. 96 с.
6. Орельен Жерон. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем: книга. Москва: Вильямс, 2018. 688 с.
7. Джулли А., Пал С. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow: книга. Москва: ДМК-Пресс, 2018. 287 с.
8. Буэно Г. Г., Суарес О. Д., Эспиноса Х. Л. Обработка изображений с помощью OpenCV: книга. Москва: ДМК-Пресс, 2016. 210 с.
9. Tutorial: Create your first Android application. URL: https://www.jetbrains.com/help/idea/create-your-first-android-application.html (дата звернення: 13.09.2021).
10. Cuda FAQ. URL: https://developer.nvidia.com/cuda-faq (дата звернення: 15.09.2021).

11. Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов: книга. Санкт-Петербург: Big Nerd Ranch, 2021. 704 с.
12. Настроить Android Emulator в Android Studio. URL: https://betacode.net/10413/configure-android-emulator-in-android-studio (дата звернення 17.09.2021).
13. Эмулятор Android. URL: https://android-tools.ru/coding/emulya-tory-v-android/ (дата звернення: 18.09.2021).
14. Создание AVD в Android Studio. Эмулятор устройства. URL: https://dimlix.com/adv-android-studio/ (дата звернення: 23.09.2021)
15. Гриффитс Д., Гриффитс Д. Основы Kotlin: книга. Санкт-Петербург: СПб, 2020. 464 с.
16. Documentation for app developers. URL: https://developer.android.com/docs (дата звернення: 27.09.2021).