УДК 004.032.2

Я. С. Парамуд, В. І. Яркун
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

МЕТОД РОЗПІЗНАВАННЯ СИМВОЛІВ НА ЗОБРАЖЕННІ НА ОСНОВІ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Розроблено метод розпізнавання тексту - рукописного чи друкованого - на зображенні. Метод грунтується на емпіричному опрацюванні зображень у статистичних моделях машинного навчання та функціонування. Він забезпечує ефективне розв’язання задач двох класів: виявлення тексту на зображенні та розпізнавання тексту. Розроблено алгоритмічні підходи, що об’єднують ці два класи задач. Алгоритмічно-програмні засоби створено та протестовано для операційної системи iOS 11.0 та нових пристроїв компанії Apple – iPhone, iPad, які підтримують цю операційну систему. Емпірично встановлено, що запропонований метод та розроблені засоби можуть бути застосовані у нових пристроях компанії Apple: iPhone, iPad, які підтримують основні особливості операційної системи iOS.

Ключові слова: розпізнавання символів, зображення, згорткова нейронна мережа, машинне навчання.

Y. Paramud, V. Yarkun
Lviv Polytechnic National University,
Computer Engineering Department

METHOD OF IMAGE SYMBOL RECOGNITION ON THE BASIS OF CONVOLUTIONAL NEURAL NETWORK

In this article, a system of handwritten or printed text recognition in the image has been developed. Empirical methods of image processing and statistical models of machine learning and simulation are being developed in two directions: the detection of text on the image and the recognition of the text. Thus, in this paper, algorithmic software tools that combine these two areas in the software created for the operating system iOS 11.0 or later for devices of the company Apple – iPhone, iPad that support this operating system are developed.

Key words: character recognition, image, convolutional neural network, machine learning.

Вступ

Дослідники, які займаються виявленням та розпізнаванням тексту на зображенні та відео, намагаються розвинути комп’ютерну систему з можливістю автоматично зчитувати текстовий вміст із зображень чи відео зі складним фоном. Загальна комп’ютерна система розпізнавання тексту повинна давати відповіді на два питання: “де знаходиться текст на зображенні?” і “що цей текст означає?” Інакше кажучи, використовуючи таку систему, текст повинен бути автоматично виявлений і кожен символ може бути розпізнаний.

Безупинне вивчення та вдосконалення виявлення і розпізнавання тексту мотивується сучасним станом та широкою областю застосування цифрового мультимедіа. Сьогодні все більше

візуальної та аудіоінформації збирають, зберігають, доставляють та керують нею у цифрових формах. Використання пристроїв мультимедіа, а саме смартфонів, які сьогодні дуже поширені у повсякденному житті, призводить до появи нових викликів та до керування великими мультимедійними базами даних. Можливість розпізнавання передбачає схожість однотипних об'єктів. Незважаючи на те, що всі предмети і ситуації унікальні в строгому сенсі, між деякими з них завжди можна знайти схожість за тією або іншою ознакою. Звідси виникає поняття класифікації – розбиття всієї множини об'єктів на непересічні підмножини – класи, елементи яких мають деякі схожі властивості, що відрізняють їх від елементів інших класів.

Окреслення проблеми

Розроблення методів машинного розпізнавання дає змогу розширити коло виконуваних комп'ютерами завдань і зробити машинну переробку інформації більш інтелектуальною. Прикладами сфер застосування розпізнавання можуть слугувати системи розпізнавання тексту, машинний зір, розпізнавання мови, відбитків пальців та інше. Незважаючи на те, що деякі з цих завдань вирішує людина на підсвідомому рівні з великою швидкістю, сьогодні ще не створено комп'ютерних програм, які вирішують їх у настільки ж загальному вигляді. Існуючі системи призначені для роботи лише в спеціальних випадках зі строго обмеженою сферою застосування. Тому створення програмних засобів для розпізнавання рукописного тексту та дослідження цієї області є актуальним.

Аналіз останніх досліджень та публікацій

Через те, що вартість обробки зображень чи відео доволі висока та вимагає значних затрат з боку процесора, більшість мобільних додатків залежать від даних, інтерпретованих у цифровому вигляді. Дані повинні бути проструктуровані, щоб вони могли бути легко опрацьовані комп’ютером. Також важливо, щоб дані були релевантні та доступні для пошуку, наприклад, у текстовому форматі. Щоб автоматично чи інтерактивно отримувати такі дані, необхідно виконувати пошук за мультимедійними базами даних, використовуючи описові функції (особливості), які стосуються об’єкта дослідження: фотографій, відео тощо [3, 4].

Особливості, такі як колір, текстура, форма, рух, макет, можна отримати під час обробки на низькому рівні, використовуючи технології комп’ютерного бачення. Різновиди особливостей, які стосуються кольору, використовують як атомні блоки для пошуку зображень та відео, наприклад, домінуючий колір, регіональна чи глобальна гістограма, вектори усередненого кольору та власне зображення. Особливості текстури, такі як гістограми орієнтації краю, параметри випадкових полів Маркова, коефіцієнти перетворення вейвлет та Фур’є, локальні бінарні шаблони також широко використовуються для індексування природних образів.

Особливості форми переважно використовують в задачах пошуку та моделювання об’єктів. Типовими особливостями форми є область накладання, параметри перетворення Хафа, елементарні дескриптори, вимірювання зміни кривизни простору, дескриптори Фур’є та кути країв. Особливості руху під час обробки на низькому рівні містять величину руху та похідні. Особливостями макета можуть бути особливості просторових вимірювань у зображеннях або в кадрах відео.

Особливості низькорівневої обробки можна легко отримати, проте це не дасть чіткого уявлення про те, що власне присутнє на зображенні. Для отримання більш описових функцій особливості низькорівневої обробки переважно зливаються в більш інформативні об’єкти та події, наприклад, текст, людське обличчя, велосипед, будинок, небо, хмари, сонце, пляж тощо. Виявлення та розпізнавання об’єктів високого рівня привертає все більше уваги науковців. Наприклад, різні способи виявлення та розпізнавання обличчя були відомі протягом багатьох років і демонструють доволі вагомі результати.

Незважаючи на це, розпізнавання інших об’єктів, зокрема тексту, залишається актуальною проблемою і сьогодні [4–6].

Існує два види тексту на зображеннях: текст сцени та накладений текст. Текст сцени – це ті текстові рядки, які написані на деяких об'єктах у зображеннях. Вони, як правило, мають великі розміри, але можуть мати оклюзії, різні вирівнювання та рухи. Накладений текст не має оклюзій, як правило, є малого розміру та релевантний до відповідного зображення.

Мета статті

Метою цієї роботи є розроблення та дослідження алгоритмічно-програмних засобів розпізнавання тексту із використанням згорткової нейронної мережі (CNN). Текст для розпізнавання може бути як писаний, так і друкований. Розробити програмне забезпечення для реалізації та демонстрації цих засобів розпізнавання тексту.

Основний матеріал дослідження

Для побудови алгоритмічно-програмних засобів розпізнавання рукописного тексту слід поділити роботу системи на чотири етапи.

Спочатку система повинна виконати задачу виявлення тексту на зображенні, де результатом можуть бути певні координати тексту чи символів на зображенні. Наступним етапом буде обробка зображення. На цій стадії роботи системи можна виконувати нарізку символів із зображення для подачі на розпізнавання. Останніми етапами є виділення ключових особливостей, які будуть використовуватись на момент розпізнавання символів, та власне розпізнавання символів [1–4].

На рис. 1 зображено основні засоби для побудови системи розпізнавання рукописного тексту, де на вході система отримуватиме певне зображення, а на виході демонструватиме результат оцифрованого документа.

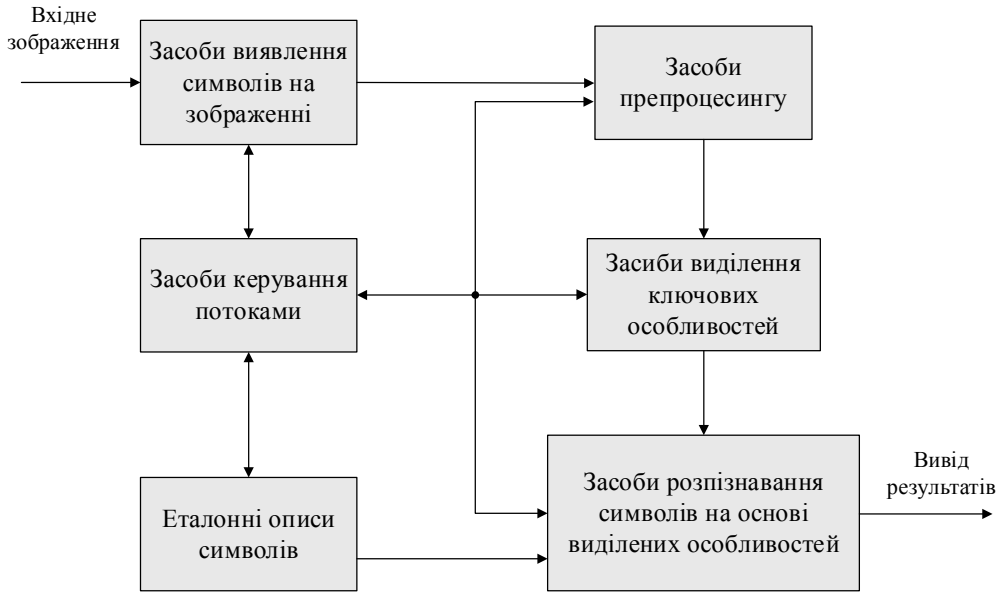

Рис. 1. Структура побудови алгоритмічно-програмних засобів розпізнавання рукописних текстів

Перш ніж почати розробляти систему, слід скласти план роботи та побудови системи, що, своєю чергою, допоможе зрозуміти суть поставленої задачі. Якщо в етапах розроблення будуть присутні певні незалежні завдання, то їх можна легко побачити та розділити роботу, що пришвидшить реалізацію системи.

На рис. 2 зображено загальну структурну схему етапів розроблення системи розпізнавання рукописного тексту. Спочатку слід реалізувати модуль виявлення тексту, використовуючи фреймворк Vision. Після цього реалізують логіку нарізки символів, використовуючи вихідні дані з Vision.

Рис. 2. Структурна схема етапів реалізації системи

Потім реалізують модель розпізнавання рукописного тексту, яку розроблено мовою Python з використанням бібліотеки tensorflow. Також важливим етапом є пошук даних для тренування, бо у випадку їх відсутності система не працюватиме – неможливо натренувати нейронну мережу без прикладів. Потім відбувається безпосереднє тренування мережі та експортування натренованої моделі у формат .mlmodel. Після цього можна підключати цю модель до проекту та використовувати для розпізнавання нових вхідних символів. Останній, але не менш важливий етап – це реалізація загальної логіки програми, яка об'єднуватиме всі програмні модулі та виводитиме результат розпізнавання користувачу.

Першим етапом розроблення системи є реалізація модуля виявлення рукописного тексту. Для цього варто спочатку підключити фреймворк Vision та імпортувати його у відповідний клас-модуль, який відповідатиме за виявлення тексту. Після чого можна почати розроблення самого модуля.

Спочатку слід створити запит (request) для відправки його на обробку у Vision – так, як це відображено у підпрограмі 1:

```
lazy var textDetectionRequest: VNDetectTextRectanglesRequest = {
    let request = VNDetectTextRectanglesRequest(completionHandler:
self.handleTextDetectionCompletion)
request.reportCharacterBoxes = true
    return request
  }()
```

Запит відповідає типу даних VNDetectTextRectanglesRequest, що дає фреймворку зрозуміти, що робота полягатиме у виявленні тексту на зображенні.

Тіло функції handleTextDetectionCompletion відображено у підпрограмі 2:

```
func handleTextDetectionCompletion(request: VNRequest, error: Error?) {
    guard let observetions = request.results as? [VNTextObservation] else {
       fatalError("unexpected result type from VNDetectTextRectanglesRequest")
    }
    self.process(observations: observetions)
  }
```

Функція handleTextDetectionCompletion відповідає за обробку відповіді від фреймворку Vision, який повертає результати типу VNTextObservation, що містять координати розташування символів на зображенні. Запускає обробку зображення функція processImage(), яку відображено у підпрограмі 3:

```
func processImage() {
    guard let ciImage = CIImage(image: image) else { return }
    let handler = VNImageRequestHandler(ciImage: ciImage, orientation: image.convertToCGOrientation())
    guard let _ = try? handler.perform([textDetectionRequest]) else {
      print("Handle request error")
      return
    }
  }
```

Спочатку функція конвертує зображення у тип даних CIImage для покращення результатів обробки зображення, що було виявлено експериментально. Після цього створюється обробник запитів (handler), який запускає раніше створений запит на обробку із вхідним зображенням, яке слід опрацювати. Після успішного виконання запиту відбудеться виклик функції handleTextDetectionCompletion по зворотньому зв'язку. Ця функція повинна обробити відповідь раніше викликаного запиту, тому буде виконуватись підпрограма 4:

```
private func process(observations: [VNTextObservation]) {
    characters.removeAll()
    let imageRect = CGRect(origin: .zero, size: image.size)
    var rectsToCrop: [CGRect] = []
    for observation in observations {
      guard let characterBox = observation.characterBoxes else { return }
      for character in characterBox {
        let transformedRect = character.boundingBox.transform(to: imageRect)
        rectsToCrop.append(transformedRect)
      }
    }
    DispatchQueue.main.async {
    self.cropCharacters(using: rectsToCrop)
    self.completion?()
    }
  }
```

У відповідь на запит буде отримано координати та розміри символів, які можна передати на подальшу обробку, що відбувається на наступному етапі – нарізка символів. Головну функцію нарізки символів виконує підпрограма 5:

```
func crop(to rect: CGRect) -> UIImage? {
    UIGraphicsBeginImageContextWithOptions(rect.size, false, self.scale)
    let origin = CGPoint(x: rect.origin.x * CGFloat(-1), y: rect.origin.y * CGFloat(-1))
    self.draw(at: origin)
    let result = UIGraphicsGetImageFromCurrentImageContext()
    UIGraphicsEndImageContext();
    return result
  }
```

Наступним є етап створення моделі для розпізнавання нейронної мережі. Основою цього етапу є створення функції тренування (train), яка зберігатиме натреновану модель як готовий продукт для подальшого використання. Функцію тренування знаходять за посиланням.

Для побудови та тренування моделі використовують бібліотеку tensorflow. Отже, згорткова нейронна мережа складатиметься з трьох згорткових шарів, де на 1 шарі кількість фільтрів буде 32, на

другому 64, на третьому 128. Величина кроку під час операції згортки дорівнює 2 на кожному з цих шарів. Як функцію активації використано функцію ReLU [6, 7]. Одним з найважливіших етапів є пошук даних для тренування мережі. В цьому випадку використано дані з ресурсів NIST [8]. Для тренування обрано 1000 екземплярів для кожного символу, для валідації – 100.

Коли модель створено і отримано всі дані для тренування мережі, починається етап тренування. Тренування відбувалось на комп'ютері MacBookPro (13-inch, Mid 2014), характеристики якого:

Processor: 2,6 GHz Intel Core i5

Memory: 8 GB 1600 MHz DDR3

Graphics: Intel Iris 1536 MB

Операційна система – macOS High Sierra (Version 10.13.4).

Для спостереження графіка тренувань використано сервіс tensorboard, в якому можна спостерігати за процесом, точністю та перехресною ентропією. Крок ітерації тренування – 1600. Тренування тривало 49 хвилин, максимальна точність, якої досягнуто, була 92 % під час тренування і 78 % під час перевірки.

Для опису роботи системи створено UML діаграми, які демонструють можливі випадки та роботу головних процесів реалізованого програмного забезпечення. Діаграму прецедентів зображено на рис. 3.

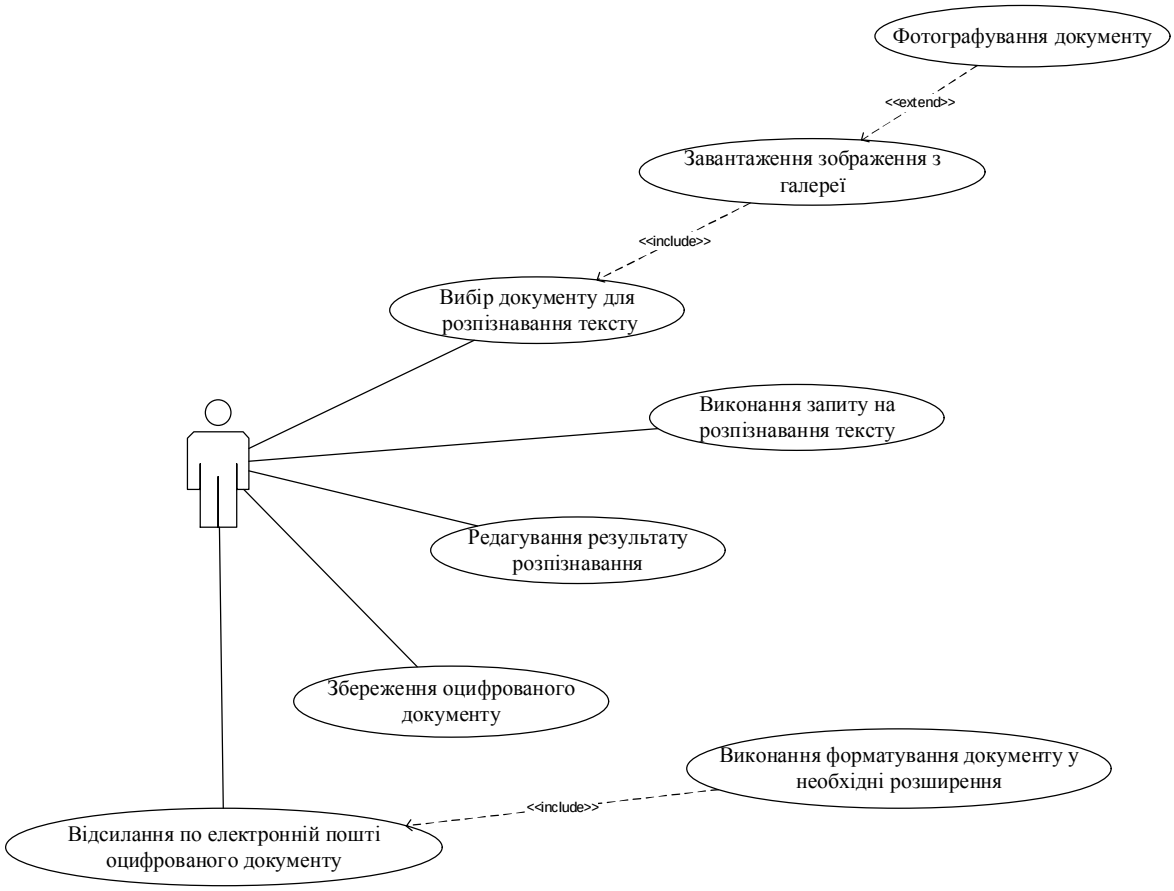

Рис. 3. Діаграма прецедентів

Користувач спочатку повинен вибрати зображення документа, який потрібно розпізнати. Він це може зробити, завантаживши існуюче зображення з галереї або сфотографувати і тоді завантажити його.

Після цього користувач виконує запит на розпізнавання, натиснувши на кнопку початку обробки, таким чином розпочинаючи процес розпізнавання. Коли процес розпізнавання завершився, користувач буде бачити на екрані смартфону оцифрований текст документа. Цей текст відображатиметься в інтерактивному вікні, дозволяючи користувачу його редагувати за необхідності.

Після завершення процесів розпізнавання та редагування користувач може зберегти оцифрований документ або відіслати його електронною поштою, за потреби форматуючи його у необхідний формат. Детальні кроки виконання програми можна побачити на рис. 4, де зображено діаграму послідовності системи.

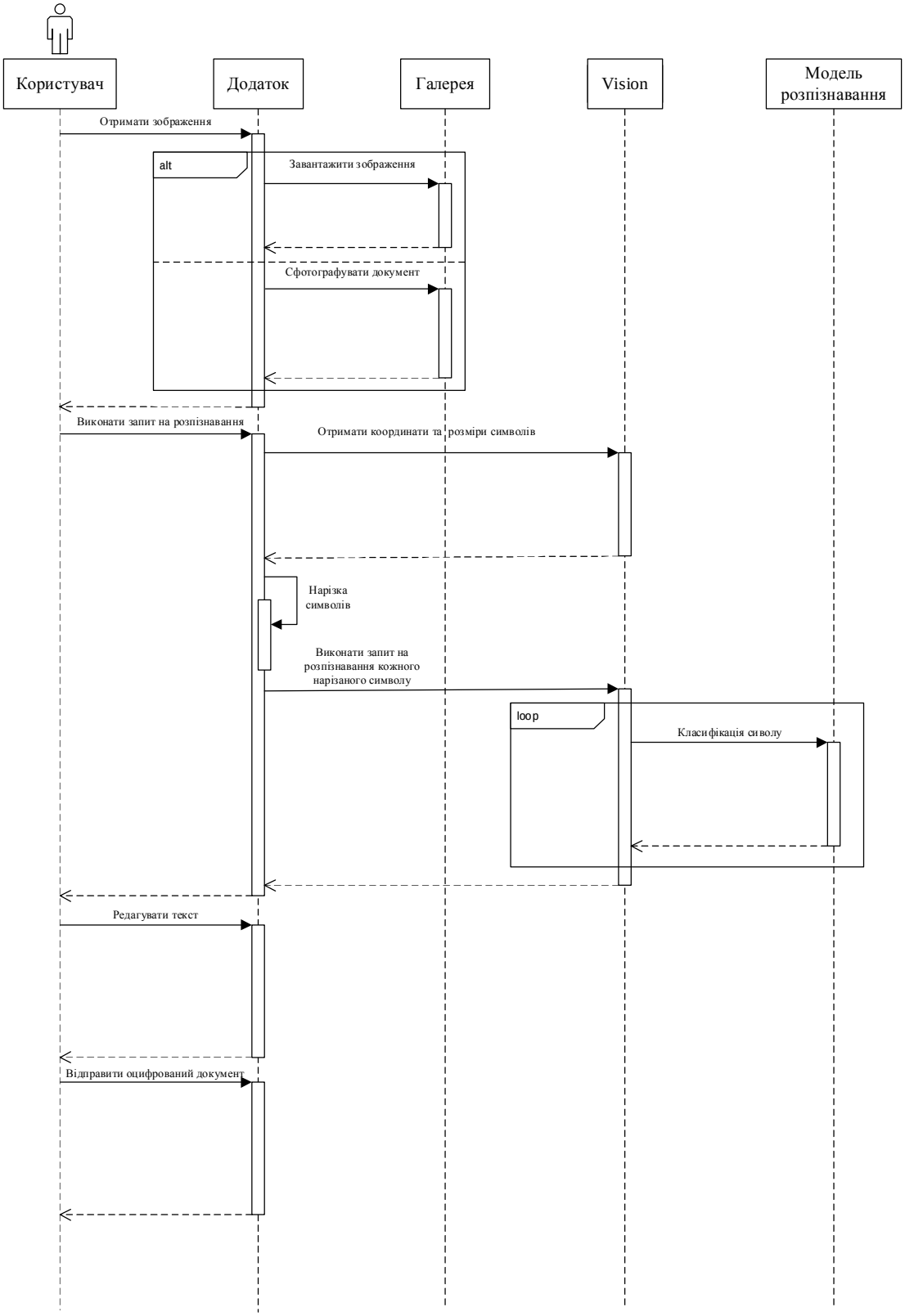

Рис. 4. Діаграма послідовності

За діаграмою послідовностей можна зрозуміти взаємодію об’єктів, впорядкованих за часом. Користувач спочатку має обрати документ для розпізнавання; він це може зробити, обравши зображення з галереї або сфотографувавши його. Після цього користувач зобов’язаний дати команду старту роботи додатка над зображенням. Додаток звертається до фреймворку Vision для отримання координат та розміру символів на зображенні, після чого додаток виконує нарізку символів. Результатом нарізки буде масив зображень, які треба розпізнати; для цього, використовуючи фреймворк Vision, формують запит для звернення до моделі розпізнавання символів, яка на вхід отримуватиме по одному символу і класифікувати його; цей процес буде циклічним, поки всі символи масиву не пройдуть класифікацію. Після розпізнавання користувач вже зможе побачити оцифрований документ і його відредагувати і за необхідності відправити комусь електронною поштою.

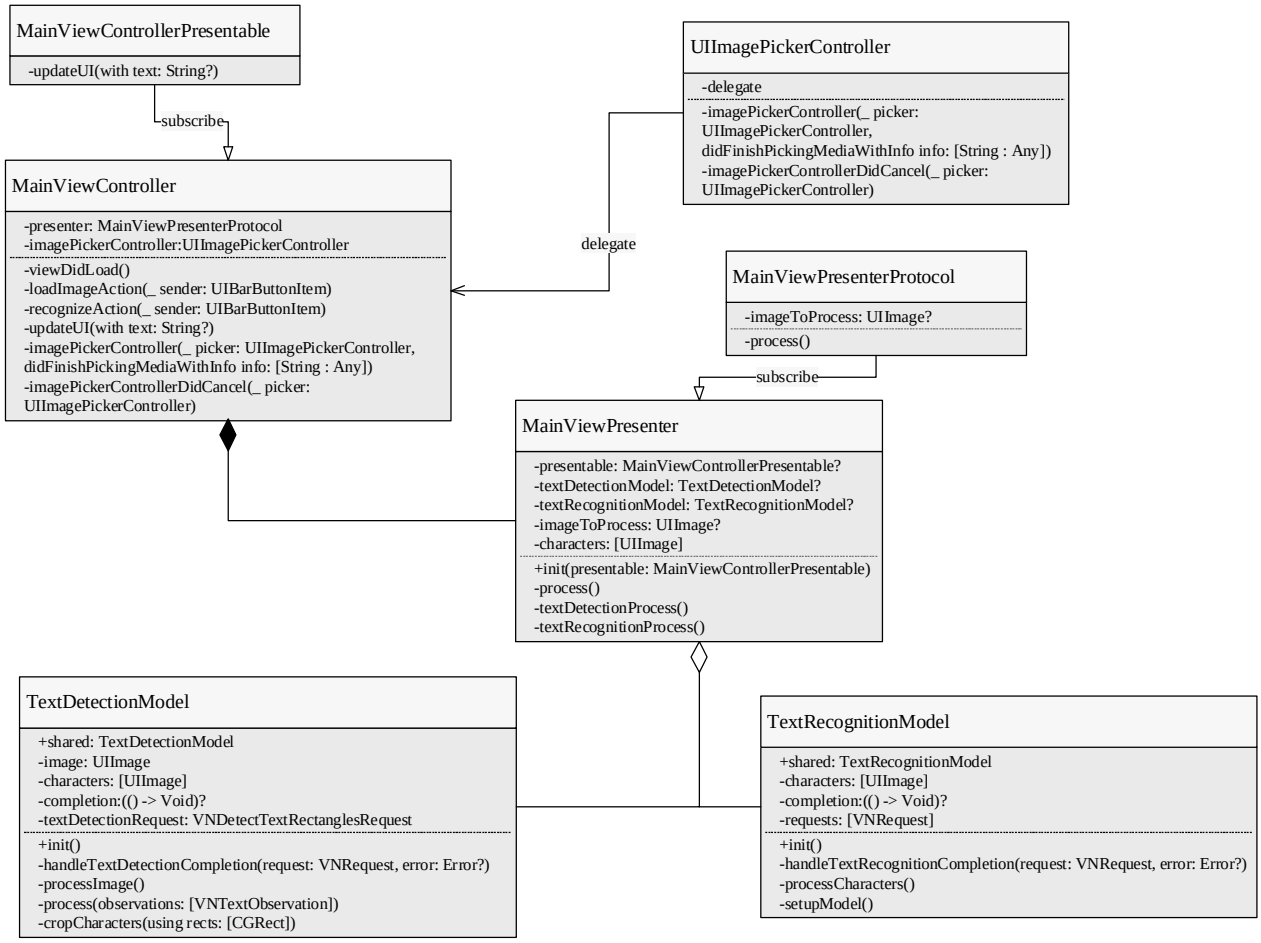

Рис. 5. Діаграми класів

На рис. 5 зображено діаграму класів, де можна побачити архітектуру програмного додатка та зв’язки між різними компонентами. Робота системи починається з класу MainViewController, який підписується на протокол MainViewControllerPresentable. Через цей протокол MainViewPresenter зможе отримати доступ до певного функціоналу MainViewController. Також MainViewController є делегатом класу UIImagePickerController, який використовується для доступу до галереї та вибору зображення на розпізнавання. MainViewControllerє композитом для MainViewPresenter. Якщо він знищиться і не існуватиме, то так само знищиться і MainViewPresenter. MainViewPresenter підписується на протокол MainViewPresenterProtocol для того, щоб MainViewController мав доступ до певного функціоналу MainViewPresenter. MainViewPresenter агрегує функціонал на виявлення та розпізнавання тексту у два модулі: TextDetectionModel та TextRecognitionModel. TextDetectionModel відповідає за виявлення тексту на зображенні, використовуючи фреймворк Vision, та нарізку

символів. TextRecognitionModel виконує розпізнавання символів, використовуючи натреновану модель, та повертає результати класифікації символів.

Архітектура, зображена на рис. 5, називається MVP, де можна виділити клас Model, який реалізовує певну логіку роботи програми, View, яка відображає користувачу результати роботи класу Model, Presenter, який використовується як комунікатор між попередніми двома компонентами.

Експериментально досліджували алгоритмічно-програмні засоби у два етапи. Спочатку тестували виявлення тексту на зображенні та нарізку символів. Другий етап – це тестування моделі розпізнавання, яке відбувалось у середовищі програмування мови Python. Отже, спочатку тестують виявлення тексту. Для цього створено мобільний додаток мовою програмування Swift у середовищі Xcode для операційної системи iOS 11.0.

а *б*

Рис. 6. Галерея зображень

Для початку тестування необхідно вибрати зображення – для цього натискають на кнопку Load Image. При натисканні кнопки Load Image відкриється галерея зображень телефону (для тестування було вибрано рукописний англійський алфавіт, рис. 6, *а*). Після вибору зображення буде знову відкрито головне вікно додатка. Для початку роботи над зображенням слід натиснути на кнопку Detect characters.

На рис. 6, *б* продемонстровано результат нарізки символів із зображення. Наступним етапом тестування буде тестування моделі розпізнавання. Модель розпізнавання тестують у середовищі програмування мови Python. Тому спочатку необхідно відкрити у терміналі вікно для виконання команд мовою Python – це можна зробити ввівши команду python або python3. Після цього тестування супроводжуватимуть такі команди:

```
>>>import coremltools
>>>import numpy as np
>>>model=coremltools.models.MLModel('/Users/AdamFox/Desktop/Models/Recognition_model.mlmodel')
>>> from PIL import Image
>>>im=Image.open('Users/AdamFox/Projects/ScienceWork/TeRec_Python/Dataset/A/hsf_7/hsf_7_00067.png')
>>>pred = model.predict({'input__input__0': im})
>>>pred = pred['logits__0']
>>> np.flip(np.argsort(pred), 0)
```

Результат відображає масив посортованих індексів, де кожному індексу відповідає певний символ. Індекси 0–9 відповідають цифрам, які дорівнюють цим індексам, індекси 10–35 відповідають великим буквам англійського алфавіту у відповідному порядку, індекси 36–61 – маленьким буквам англійського алфавіту відповідно. Для тестування було використано зображення з великою буквою «A», перший елемент масиву дорівнює 10, що є коректним індексом, який відповідає цій букві.

Для ще однієї перевірки було введено такі команди:

```
>>>im=Image.open('Users/AdamFox/Projects/ScienceWork/TeRec_Python/Dataset/B/hsf_1/hsf_1_
00002.png')
>>>pred = model.predict({'input__input__0': im})
>>>pred = pred['logits__0']
>>> np.flip(np.argsort(pred), 0)
```

Результатом буде знову масив, але з першим елементом 11. Цей індекс відповідає букві «B», яка відправлялась на розпізнавання, отже, результат є коректним.

Висновки

Проаналізовано існуючі проблеми створення систем розпізнавання символів на зображенні. Розроблено метод розпізнавання тексту – рукописного чи друкованого – на зображенні із використанням згорткової нейронної мережі. Продемонстровано ефективне розв'язання задач двох класів: виявлення тексту на зображенні та розпізнавання тексту. Розроблено алгоритмічні підходи, що об'єднують ці два класи задач. Для опису роботи системи створено UML – діаграми, які демонструють можливі випадки та роботу головних процесів реалізованого програмного забезпечення.

Ці засоби розпізнавання використовують локально на мобільному пристрої, що при цьому їх виділяє серед аналогів, оскільки цей підхід не вимагає підключення пристрою до мережі Інтернет та виконання завдання розпізнавання, використовуючи мережу. Це також збільшить швидкодію, тому що затримки, які виникають при виконанні запитів через мережу, будуть відсутні.

Алгоритмічно-програмні засоби створено та протестовано для операційної системи iOS 11.0 та нових пристроїв компанії Apple – iPhone, iPad, що підтримують цю операційну систему. Досягнуто 94 % середньої статистичної точності у розпізнаванні. Емпірично встановлено, що запропонований метод та розроблені засоби можуть бути застосовані у нових пристроях компанії Apple: iPhone, iPad, які підтримують основні особливості операційної системи iOS.

1. Paramud Y., Yarkun V. Algorithmic and software means of handwritten symbol recognition // Bulletin of the National University "Lviv Polytechnic" "Computer Systems and Networks" No 881, Lviv, Ukraine. 2017. – P. 98–106. 2. Yarkun V., Paramud Y. Algorithmic and software synchronization of information exchange // Bulletin of the National University "Lviv Polytechnic" "Computer Systems and Networks" No 857, Lviv, Ukraine. 2016. – P.111–118. 3. Datong Chen, Jean-Marc Odobez, Herve Bourlard. Text detection and recognition in images and video frames // Pattern Recognition journal, 2003. – P. 595–608. 4. Convolutional Neural Network [Electronic resource]/ Stanford. – Access mode: http://ufldl.stanford.edu/tutorial /supervised/ConvolutionalNeuralNetwork/. 5. Convolutional networks [Electronic resource]/ Stanford. – Access mode: http://cs231n.github.io/convolutional-networks/#overview. 6. Coreml [Electronic resource]/ Apple. – Access mode: https://developer.apple.com/documentation/coreml. 7. Bachelor's degree dissertation [Electronic resource]/ Github. – Access mode: https://github.com/VitaliyYarkun/Bachelor-s-degree-dissertation. 8. NIST. American national standard for information systems – data format for the interchange of fingerprint, facial, and, scar mark and tattoo (smt) information, ansi-itl 1-2000 (nist special publication 500–245), September 2000.