МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри

_____ Жуков І. А.
(підпис) (ПІБ)

"_____" _____ 2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Комп'ютерна система розпізнавання зображень»

Виконавець: студент групи КС-201Мз

Ярош Євгеній Вікторович _____
(підпис)

Керівник: к.т.н., доцент
Малярчук Василь Олександрович _____
(підпис)

Нормоконтролер: _____ В.І.Анреєв
(підпис)

Засвідчую, що у магістерській роботі немає запозичень праць інших авторів без відповідних посилань
Студент_____Є.В. Ярош

Київ 2020

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Напрямок (спеціальність):123 "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Жуков І. А.
(підпис) (ПІБ)
" " _____ 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи
Яроша Євгенія Вікторовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи «Комп'ютерна система розпізнавання зображень»

затверджена наказом ректора від " 02 " жовтня 2020 р., №2258/ст

2. Термін виконання роботи: з "23" серпня 2020 р. по "30 " грудня 2020 р.

3. Вихідні дані до роботи: технологія Web, апаратні та програмні засоби створення інформаційних систем

4. Зміст пояснювальної записки:

1) дослідження системи розпізнавання зображень як об'єкту застосування інформаційних систем

2) порівняльний аналіз існуючих систем розпізнавання зображень

3) огляд апаратного забезпечення

4) вибір інструментів та технологій розробки

5)проєктування та розробка програмного забезпечення системи розпізнавання зображень

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

презентація *MS Power Point.*

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Отримання завдання на дипломну роботу.	23.09.2020 – 25.09.2020	
2.	Огляд та порівняння існуючих систем. Написання розділу 1.	26.09.2020 – 15.10.2020	
3.	Огляд апаратного забезпечення системи	16.10.2020 – 31.11.2020	
4.	Вибір обладнання та системи. Написання розділу 2.	01.11.2020 – 12.11.2020	
6.	Вибір інструментів та технологій розробки. Написання розділу 3	13.11.2020 – 24.11.2020	
7.	Проєктування програмного забезпечення системи	25.11.2020 – 30.11.2020	
9.	Розробка програмного коду продукту. Написання розділу 4	01.12.2020 – 07.12.2020	
10.	Оформлення текстових матеріалів дипломного проєкту. Створення презентації	08.12.2020 – 30.12.2020	

7. Дата видачі завдання: "23 " серпня 2020 р.

Керівник дипломної роботи _____ Малярчук В.О.
(підпис керівника) (ПІБ.)

Завдання прийняв до виконання _____ Ярош Є.В.
(підпис випускника) (ПІБ.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Комп’ютерна система розпізнавання зображень» містить: 82 с., 28 рис., 3 табл., 22 літературних джерела.

Об'єкт дослідження: комп'ютерна система розпізанвання образів.

Мета роботи: Проєтування та розробка системи розпізнавання зображень.

Предметом дослідження: методи та підходи для розпізнавання рукописного тексту.

Методи дослідження: алгоритми та підходи для вирішення задач розпізнавання тексту з зображення.

Практичне значення: В результаті проведення порівняльного аналізу існуючих метоів було прийнято рішення взяти все найкраще з існуючих метоів й втілити їх в реальність, що й продемонстровано в роботі.

Реалізована у роботі система з використанням Web-технологій має досить низьку вартість, робота системи потребує мінімальної зміни апаратного і програмного забезпечення. Таким чином ця система може бути використана для проєктів з низьким бюджетом.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ОРТ – оптичне розпізнавання тексту

OCR – optical character recognition

ШНМ – штучна нейронна мережа

ПХ – перетворення Хафа

ЗМІСТ

ВСТУП

Пройшли ті часи, коли всі документи організацій, книжки, форми, довідки і т.д. зберігалися тільки в паперовому або друкованому вигляді. В останні десятиріччя перед суспільством постала проблема переведення тексту друкованих книжок в більш зручний для 21 сторіччя цифровий формат. За останні 7-8 років за допомогою систем компанії «Гугл» було оцифровано більше 15 мільйонів книжок і це число росте з кожним днем.

В наші дні майже всі компанії ведуть електронний документообіг, в бібліотеках можна взяти не лише друковані підручники, а довідки в державних установах поступово починають видавати в електронному вигляді. Все це допомагає опрацьовувати різного роду документи дуже швидко та зручно вести детальний звіт по всім галузям.

В таких випадках за великої кількості документів на паперових носіях необхідно перевести такі документи в електронний вигляд. Але переводити усі документи не в автоматизованому порядку дуже складно та дуже затратно. Тому необхідно впроваджувати та використовувати розпізнавання усіх видів документів, починаючи від чеків і закінчуючи багатотомними паперовими примірниками керування ракетами.

На перший погляд, може здатися, що це настільки несуттєва автоматизація, що не треба їй приділяти занадто багато часу. Але це не так. За останніми дослідженнями середньостатистична компанія в середньому втрачає близько 10 відсотків свого прибутку тільки через ту причину, що хтось із співробітників при ручному введенні документів допускав досить суттєві помилки. Іншою шокуючою цифрою є те, що 40 відсотків робочого часу витрачається лише на пошук оригіналу документу (в паперовому вигляді), якщо в компанії відсутній електронний документообіг. І що вже казати про те, що більшість документів є тим чи іншим типом форм, тобто структурованим документом чітко заданої форми. Враховуючи цифри, наведені вище, можна оцінити навіть в грошовому

еквівалентні вигоду від впровадження системи розпізнавання різного роду документів, форм чи, навіть, книжок.

Переведення друкованого (написаного) тексту в цифровий формат здійснюється за допомогою систем оптичного розпізнавання символів (optical character recognition – OCR). Система здійснює поетапне оцифрування і розпізнавання кожного символа, що отримує на вході

Актуальність теми полягає в тому, що комп’ютерний зір – це сучасний напрямок технологій який має широкий потенціал для використання у великій кількості сфер діяльності. Перспективним напрямом є саме розпізнання друкованого тесту та його частин. Такі технології могли б широко використовуватися у медицині, системах відеоспостереження або у кіноіндустрії, щоб не використовувати дорогі прилади захоплення руху.

В даній дипломній роботі в якості **об'єкта дослідження** вибірка цифрових знімків, що містять друкований та рукописний текст.

Предметом дослідження є методи та підходи для розпізнавання друкованого та рукописного тексту.

Мета роботи

Розробка системи для розпізнавання рукописного і друкованого тексту.

Методи дослідження

Алгоритми та підходи для вирішення задач розпізнавання тексту з зображення.

Наукова новизна одержаних результатів та їх практичне значення

Розроблена програма проста у використані і задовольняє всі потреби користувача. Її завжди можна мати під рукою – відкрити на комп'ютері або на смартфоні. Зовнішній інтерфейс мінімалістичний, лаконічний та зрозумілий, який також може адаптуватись до різних розмірів екранів комп'ютера, чи до версії смартфону.

Усі дії при роботі з системою, такі як, зчитування, додавння та збереження - інтуїтивно зрозумілі.

Реалізований у роботі додаток з використанням Web-технологій має досить низьку вартість, робота системи потребує мінімальної зміни апаратного і програмного забезпечення. Таким чином ця система може бути використана для проєктів з низьким бюджетом.

РОЗДІЛ 1
ЗАГАЛЬНИЙ АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ТЕКСТУ

1.1. Еволюція підходів призначених для оптичного розпізнавання тексту

Наприкінці 20-х років. Густав Тучек розробив у Німеччині патент на метод оптичного розпізнавання тексту (ОРТ). Кілька років потому Гендель приєднався до галузі разом з ним, і в 1 із 33 він аналогічним чином скасував патент у США. У 1935 році Точек отримав ще один американський патент, подібний до свого попередника. Його розробкою є пристрій, що використовував ідентифікаційні шаблони [1].

У середині 20 століття Девід Х. Шепард, аналітик, який добре розбирається в шифруванні Агентства безпеки збройних сил США, проаналізував проблему форматування повідомлень, що вводяться у двійковий код для комп'ютерної обробки [1]. Потім він створив систему, яка вирішує цю проблему. Він повідомив Washington Daily News (1 квітня) та New York Times (53 грудня) після отримання патенту США на розробку. Після цього Шепард створив власну компанію та створив розумні пристрої. Протягом наступних кількох років компанія створила першу у світі комерційну систему розпізнавання тексту.

Найдосконаліша система була використана в 1955 р. Для "Reader's Digest". Наступні системи були продані Stadart Oil для читання банківських систем та для належної та зручної роботи з друкованими чеками. Інші системи, що поставляються Шепардом, продавались наприкінці 1950-х із сканерами сторінок ВПС США для читання та надсилання текстових повідомлень. Протягом кількох років IBM змогла отримати деякі патентні ліцензії у Shepard [1].

Після 1960-х рр. Readers Digest і RCA спільно розробили свою власну систему читання приватних документів, яка використовує ORT для оцифровки серійних номерів купонів Readers Digest, що повертаються з рекламних оголошень. Для документів, роздрукованих на принтері RCA, був обраний спеціальний текст OCR-A [2]. Документообіг працював безпосередньо з комп'ютером RCA 301 (одним з перших суперкомп'ютерів). Швидкість комп'ютера становила 25 файлів в секунду: вона перевіряла кожен новий вхідний файл, крім тих, які не могла правильно обробити. Поштова служба США використовує машини OPT для сортування пошти з 1965 року на основі технології, розробленої дослідником Яковом Рабінова. В Європі поштове відділення Великобританії було першою публічною компанією, яка використовувала апарати оптичного розпізнавання тексту. Пошта Канади використовувала систему OPT з 1971 року до сьогоднішнього дня [3]. Спочатку в сортувальному центрі система розпізнавання зчитує ім'я та адресу одержувача і друкує штрих-код на конверті. Він наноситься спеціальним чорнилом, які потім можна побачити в ультрафіолеті. Це зроблено, щоб уникнути плутанини з особистим адресою, який може відображатися в будь-якому місці конверта.

У 1974 році новатор Рей Курцвейл відкрив свою власну компанію Kurzweil Computer Products і почав працювати над розробкою власної системи OPT, яка може розпізнавати текст у всіх шрифтах. Він вважав, що кращий спосіб привернути цю технологію - створити машину, яка може читати текст для сліпих. [3]. Комп'ютер зажадав винаходи відразу двох нових методів розробки - сканера тексту і якогось синтезатора мови, що перекладає текст в мову. Остаточний продукт був представлений 13 січня 1976 року на прес-конференції, яку провели Курцвейл і члени Національної асоціації сліпих.

У 1978 році компанія Kurzweil Computer Products почала використовувати комерційну версію комп'ютерної системи OPT [3,4]. Два роки по тому він продав своє розумне дітище Xerox, яке дійсно хотіло заробляти на системах. розпізнавання символів. Тож ця компанія невдовзі перетворилася на дочірню компанію великого монополіста «Ксерокс».

1.2. Історія винекнення

Оптичне розпізнавання символів (OCR) - це механічне або електронне перетворення фотографій або малюнків письмових, друкованих або рукописних малюнків в певну послідовність зображень, яка в подальшому має відображатися у всіх текстових програмах. Це розпізнавання часто використовується для перетворення книг і інших текстових документів в електронний формат, для спрощення за рахунок автоматизації систем комерційного обліку або для публікації текстів на веб-сторінках. Завдяки оптичному розпізнаванню тексту стає можливим редагувати документ, шукати слова чи фрази, зберігати текст в більш стислій формі, переглядати або роздруковувати текстовий документ без втрати якості, аналізувати інформацію і виконувати електронний переказ текстів, форматування або перетворення в мова. Оптичне розпізнавання символів (OCR) - одна з найбільш вивчених проблем в області розпізнавання образів, штучного інтелекту та комп'ютерного зору.

Система ORT вимагає калібрування для роботи з конкретним шрифтом; в попередніх версіях для програмування необхідно було відображати кожен символ, програма могла працювати тільки з одним шрифтом за раз. Сьогодні найбільш поширені так звані «інтелектуальні» системи, здатні розпізнавати більшість шрифтів з високим ступенем точності. Деякі з цих систем OCR можуть відновлювати початкове форматування тексту, включаючи зображення, стовпці та інші нетекстові компоненти.

Персептрон, або перцептрон (англ. Perceptron від лат. Perceptio - сприйняття) - це математична або комп'ютерна модель сприйняття інформації мозком (кібернетична модель мозку), запропонована Розенблатта в 1957 році і реалізована у вигляді електронної системи «Марк-1». У 1960 році перцептрон став однією з перших моделей нейронних мереж, а Mark-1 став першим в світі нейрокомп'ютером. Перевершуючи його простоту, що сприймає може вивчати і вирішувати досить складні завдання. Основна математична задача, яку він може вирішити, - це лінійне поділ довільних нелінійних множин, так зване забезпечення

лінійної роздільності.

Перцептрон складається всього з трьох типів елементів (рис. 1.1), а саме: сигнали, що надходять від передавачів, відправляються на відповідні елементи, а потім на відповідальні. Тобто перцептрони дозволяють створювати набір «асоціацій» між вхідними стимулами і бажаної вихідний реакцією. З біологічної точки зору такий процес можна порівняти, наприклад, з перетворенням інформації, отриманої зором, в фізіологічний відповідь рухових нейронів. Що стосується сучасної термінології перцептрони можна віднести до штучних нейронних мереж:

а) з прихованим шаром;

б) з пороговою функцією передачі;

в) При прямому поширенні сигналу.

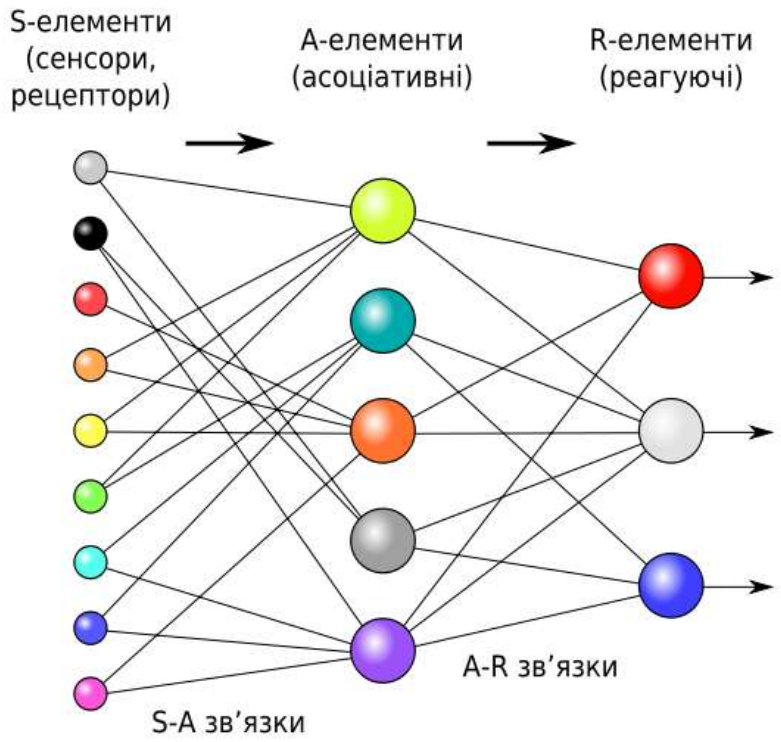

Рис.1.1. – Логічна схема перцептрону з трьома виходами

Склад елементарного перцептрона складається з трьох типів елементів: S-елемента, A-елемента і R-елемента. S-елемент - це шар датчиків або рецепторів. У фізичній формі вони можуть являти собою світлочутливі клітини сітківки або фоторезистори в матриці камери. Кожен з рецепторів може перебувати в одному з двох станів - спокою або напруги, і тільки в останньому випадку він передає

одиничний сигнал на наступний рівень, що зв'язує елементи.

А-елементи називаються асоціативними, тому що кожен такий елемент зазвичай відповідає цілому набору (асоціації) S-елементів. А-елемент активується, тільки кількість сигналів від S-елементів на його вході перевищує певне значення θ.

Сигнали від порушених А-елементів, в свою чергу, передаються на суматор R, а сигнал від і-зв'язуючого елемента передається з коефіцієнтом ω_i [5]. Цей коефіцієнт називається вагою прив'язки А-R.

Як і вентилі А, вентилі R підраховують суму значень вхідного сигналу, помножену на ваги (лінійна форма). R-елемент і, отже, елементарний персептрон, видає «1», якщо лінійна форма перевищує поріг θ, в іншому випадку вихід стає «1». Математично функцію, що реалізує R-елемент, можна записати в такий спосіб: $\boldsymbol{f(x) = sign(\sum_{i=1}^{n} \omega_i x_i - \theta)}$

Вивчення елементарного персептрона полягає в зміні вагових коефіцієнтів ланок А - R. Ваги зв'язків S-A (які можуть набувати значень (-1, 0, 1)) і значення порогових значень А-елементів вибираються випадковим чином на початку і не змінюються далі.

Після навчання перцептрон готовий до роботи в режимі розпізнавання або узагальнення [6]. У цьому положенні перцептрону представлені раніше невідомі йому об'єкти (рисунок 1.2), і він повинен визначити, до якого класу вони належать. Робота перцептрону полягає в наступному: при пред'явленні об'єкта порушені А-елементи передають сигнал R-елемента, що дорівнює сумі відповідних коефіцієнтів ω_i. Якщо ця сума позитивна, приймається рішення, що даний об'єкт належить до першого класу, а якщо негативне - до другого [7].

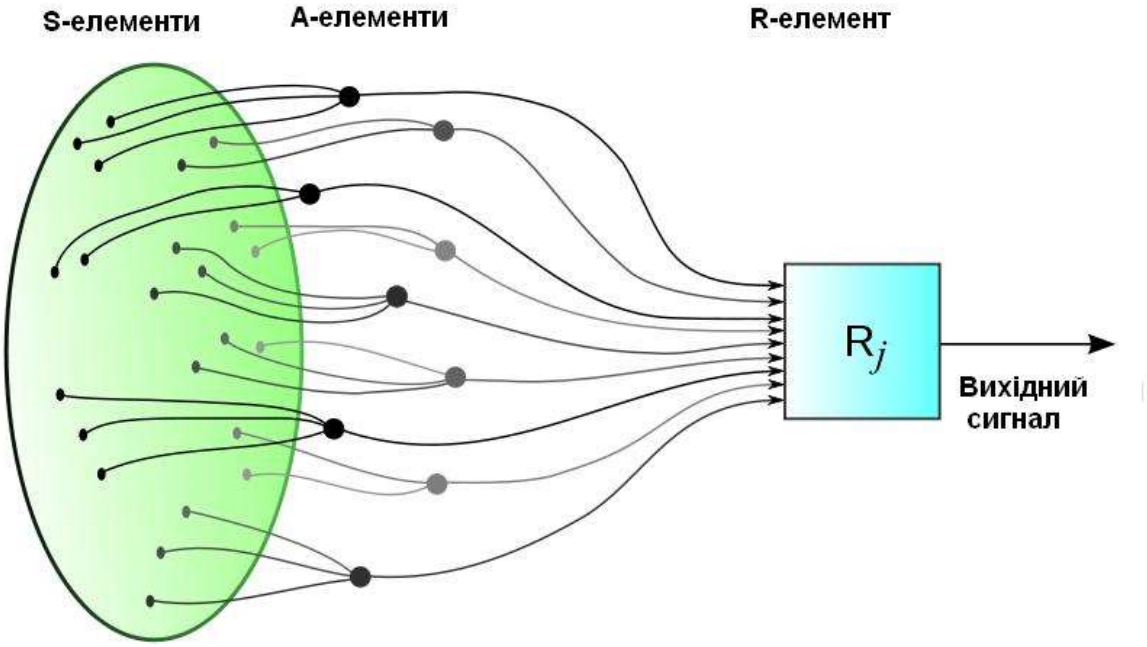

Рис.1.2. – Надходження сигналів із сенсорного поля до розв'язувальних блоків елементарного перцептрона в його фізичному втіленні

Штучна нейронна мережа (ШНМ) - це система, яка представлена у вигляді математичної моделі, за способом побудови аналогічна функцій деяких біологічних нейронних систем. Крім того, апаратна складова і програмна реалізація таких систем аналогічні принципом організації мережі нервових клітин у людей. Ця концепція виникла після вивчення певних процесів, змодельованих в людському мозку, і при спробі повторити ці процеси на практиці. Першим кроком до створення таких систем були нейрони У. Маккалок і У. Питтса. Після створення навчальних алгоритмів почали з'являтися нові моделі, стали з'являтися завдання для вирішення практичних завдань: для задач прогнозування результатів, завдань управління і завдань розпізнавання образів.

ІНС - це система пов'язаних між собою простих обробників (штучних нервових клітин), які взаємодіють один з одним [8]. Зазвичай такі нервові клітини дуже прості (особливо в порівнянні з процесорами, використовуваними в комп'ютерах користувачів). Кожен обробник в такій мережі може працювати тільки з імпульсами, які він регулярно отримує, і сигналами, які він регулярно відправляє іншим обробникам. Крім того, якщо нейрони підключені до досить великої мережі взаємодії, підлягають управлінню, то така локальна система може виконувати не

тільки прості, але і складні завдання.

З огляду на цей предмет з точки зору машинного навчання, нейронна мережа є окремим випадком методів дискримінує аналізу, кластерних методів, розпізнавання образів і т.п.

Коли справа доходить до математики, навчання нейронної мережі - це завдання нелінійної оптимізації з декількома параметрами.

З кібернетичної точки зору NM може використовуватися в задачах адаптивного управління, а також в якості алгоритмів для робототехніки.

З боку розвитку комп'ютерних технологій і програмування, за допомогою нейронної мережі можна ефективно вирішувати завдання паралелізму.

А з огляду на принципи штучного інтелекту, нейронна мережа є основою філософських непристойностей і найважливішим вектором розвитку в структурному підході до вивчення властивостей побудови (моделювання) природного інтелекту з використанням сучасних алгоритмів і методів.

Нейронні мережі не просто програмуються в звичайному розумінні цього слова. Вони вчаться. Навчання - одне з головних переваг нейронних мереж в порівнянні з усіма традиційними методами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язку між нейронами. Під час навчання нейронна мережа може виявляти складні залежності між входами і виходами і виводити узагальнення. Це означає, що нейронна мережа зможе дати нам правильну відповідь, якщо навчання виконано правильно через дані, яких не було у вихідній вибірці, або через перекручених або зашумлених даних.

Класифікація нейронної мережі по типу одержуваної інформації:

- аналогові нейронні мережі (необхідно надати інформацію у вигляді дійсних чисел);

- бінарні нейронні мережі (необхідно надати інформацію у вигляді двійкових чисел).

Класифікація за характером навчання:

- навчання з учителем - необхідні відомі результати нейронних мереж, і ми хочемо отримати аналогічні результати;

- автоматичне навчання - нейронна мережа працює тільки з вхідними даними і аналізує і відображає вихідні дані незалежно один від одного. Такі мережі зазвичай називають самоорганізації;

Навчання з підкріпленням - система нарахування нагород і штрафів в залежності від умов і навколишнього середовища.

мережі з фіксованими зв'язками (вагові коефіцієнти нейронної мережі вибираються відразу, переходять від завдання, при цьому: dV / dt = 0, де V - вагові коефіцієнти здійснювати підключення до мережі);

мережі з динамічними зв'язками (для них в процесі навчання налаштовуються сінапсовие зв'язку, тобто dV / dt ≠ 0, де V - вагові коефіцієнти для мережевих підключень).

Навчання з учителем або навчання з відомими результатами - це один з методів машинного навчання, в ході якого система суб'єкта змушена вчитися, змішуючи певну кількість наборів результатів «стимулів», щоб отримати «відповідь» на свої «стимули». який не належить до цього набору результатів ... З огляду на це питання з кібернетичної точки зору, така система є одним з певного експерименту - кібернетичного.

Мета цього підходу полягає в наступному.

Між входами і очікуваними виходами (стимулом) може бути якась невідома зв'язок. Нам відомо лише про обмежену кількість пар-попередників «стимулу», який називається навчальним тестом. На основі отриманих даних необхідно визначити приналежність (побудувати модель відносин стимулів, яка підходить для прогнозів), тобто створити алгоритм, який зможе дати щодо точну відповідь для будь-якого об'єкта. Для вимірювання точності відповідей, а також в навчанні з прикладами можна ввести функцію якості: $f(x) - e \leq y \leq f(x) + e$

Цей підхід є окремим випадком кібернетичного експерименту, але зв'язок встановлюється зворотним чином. Ця експериментальна установка включає в себе наявність конкретної системи, методу навчання, а також моделі і методу тестування системи.

Експериментальна модель в даному випадку складається з тестованої

системи, набору стимулів, отриманих із зовнішнього середовища, і системи, що контролює посилення (регулюючої внутрішні параметри). Для таких завдань ви можете використовувати автоматичний пристрій, який виконує регулювання (наприклад, термостат), або оператора (вчителі), який може обробляти відповіді тестованої системи і стимули у зовнішньому середовищі, задіюючи певні шляхи для контенту, який коригує стан така система запам'ятовування.

Є два типи: перший, якщо реакція такої системи не змінює стан навколишнього середовища, і другий, якщо система реагує на зміни зміною стимулів для навколишнього середовища. Такі підходи показують нам подібність між такими системами в порівнянні з нервовою системою людини.

Типологія контрольованих навчальних завдань

Типи введення:

а) опис функцій - ситуація, коли певний об'єкт може бути описаний окремим набором його властивостей, які називаються функціями. Ці властивості можуть бути як числовими, так і алфавітними.

б) таблиця, що показує відстань між об'єктами. Кожна така одиниця зазвичай описується відстанню до інших одиниць в навчальному тесті. Цей тип даних зазвичай обробляється певними методами, зокрема методом k найближчих сусідів, методом вікна розбору, потенційним методом роботи.

в) тимчасової ряд або сигнал - це послідовність вимірювань в часі. Кожне наступне вимір можна охарактеризувати числом, вектором і, в цілому, певним описом властивостей досліджуваного об'єкта протягом заданого періоду часу.

Є й більш цікаві випадки, коли вхідні дані надаються у вигляді текстів, графіків, результатів запитів до API і т.п. Як правило, такі символи задаються в першому або другому випадку в зв'язку з попередньою обробкою даних і витяганням функцій.

Типи оглядів:

- ситуація, коли набір можливих результатів нескінченний (висновки - числа або вектори);

- ситуація, коли є багато можливих результатів;

- ситуація, коли відповіді описують можливу поведінку явища або процесу.

Особливе значення мають дані для прогнозів і прогнози для часових рядів, для яких виділяються дані з набором певних специфічних функцій, тому їх слід класифікувати. Завдання на вивчення явищ, розвиток яких пов'язаний з цим, можна розділити на кілька класів:

За характером основних функцій об'єкта:

- Прогнозування явищ, моделі яких представлені у вигляді детермінованих часових рядів. Такі завдання можна вирішити за допомогою методів математичного аналізу;

- Прогнозування явищ, моделі яких представлені у вигляді невизначених часових рядів. Рішення цього типу завдань вирішується з використанням підходів теорії ймовірностей і теореми. статистика. Зокрема, реалізація таких явищ може приймати форму:

а) стаціонарний часовий ряд,

б) нестаціонарні часові ряди, уособлює певним еволюційним трендом в часі; при оцінці нестаціонарних процесів можна виділити області, де робота може вважатися стаціонарної. вибір інтервалу формування навчальної вибірки в цьому випадку вибирається відповідно до завдання прогнозу;

За кількістю функцій в об'єкті дослідження:

- одномірна задача;

- багатовимірна задача; об'єкт або явище представлено декількома персонажами; Завдання прогнозу можна розширити, представивши дані в просторі.

По термінах виконання розрізняються різні типи прогнозів:

- короткостроковий прогноз, Р від одного до двох;

- прогноз на перерву, Р з трьох до семи;

довгостроковий прогноз, Р від десяти до п'ятнадцяти.

Очевидно, що тип прогнозу істотно впливає на вибір методів і методів його реалізації.

1.3. Поточний стан технологій оптичного розпізнавання тексту

Досить точне розпізнавання символів в друкованому тексті сьогодні може бути досягнуто тільки з чітким зображенням, наприклад, в друкованих документах. При такій постановці завдання точність перевищує 99%, а абсолютна точність може бути досягнута тільки шляхом подальшої ручної корекції людиною. Проблема розпізнавання рукописного "друкарського» тексту і стандартного рукописного тексту, а також друкованих текстів в інших форматах (особливо з дуже великою кількістю символів) сьогодні є предметом активних досліджень.

Точність цих методів можна виміряти кількома способами, тому вони можуть сильно відрізнятися. Наприклад, якщо певне слово знайдено, його не можна знайти в словниках певного програмного забезпечення, і, таким чином, пошук неіснуючих слів може збільшити кількість помилок.

Розпізнавання символів в Інтернеті іноді можна сплутати з оптичним розпізнаванням символів. Хоча метод OCR є автономним методом розпізнавання символів, він працює зі статичної формою текстового подання, але онлайн-метод розпізнавання символів враховує рух при введенні тексту. Наприклад, метод онлайн-розпізнавання, який використовується PenPoint OS або планшетом, дозволяє вам вирішити, чи писати рядок справа наліво або зліва направо.

Останнім часом стали широко відомі такі комерційні послуги, як онлайн-системи розпізнавання рукописного тексту на льоту. Алгоритми для таких пристроїв враховують те, що порядок, швидкість і напрямок окремих ділянок вхідних ліній відомі. Крім того, користувач може використовувати тільки певні стилі письма. Ці методи не можна використовувати в програмному забезпеченні, яке використовує скановані паперові документи, тому складність розпізнавання рукописного "друкарського» тексту залишається важливою проблемою. Фотографії з рукописним «надрукованим» текстом без артефактів можуть досягти точності 80% - 90%, але з такою точністю зображення буде перетворено в текст з десятками помилок на сторінці. Ця технологія може бути корисна тільки в дуже обмеженій кількості додатків.

Розпізнавання почерку - ще одна часто вивчається проблема в сучасному світі. На сьогоднішній день точність такого позначення значно нижче, ніж у рукописного "друкарського» тексту. Висока продуктивність досягається тільки за допомогою контекстної і граматичної інформації. Наприклад, в процесі розпізнавання в словнику потрібно шукати цілі слова; це стає набагато простіше, ніж намагатися аналізувати окремі символи в тексті. У цьому випадку знання граматики мови теж може бути, це допоможе охарактеризувати слово: або це дієслово, або прикметник, або іменник. Форми деяких рукописних символів іноді можуть не містити достатньо інформації для правильного (більше 98%) розпізнавання всього рукописного тексту.

Для вирішення більш складних завдань в області розпізнавання зазвичай використовуються в основному інтелектуальні системи розпізнавання, такі як штучні нейронні мережі.

1.4. Загальний алгоритм оптичного розпізнавання тексту

Щоб система розпізнавання тексту працювала, необхідно надати їй певний цифровий файл із відсканованого тексту (текстовий елемент) як вхід. Наступним кроком у системі розпізнавання тексту є так звана "Сегментація" та попередня підготовка даних. На цьому етапі роботи коригуються елементи тексту. Елементами тексту можуть бути фотографії з певними частинами тексту, відскановані розділи сторінок та подібні варіанти розміщення цифрового тексту. Після вирівнювання всі такі елементи повинні бути видалені з непотрібних частинок, тобто текст складається виключно з символів, з якими система «знайома», а решта повинна бути очищена з поля зору системи, або просто пропустити такі елементи. Потім необхідно виконати процедуру бінаризації, яка полягає у виділенні тла та елементів, які зайвим буде читати як білі елементи, а необхідний текст вводиться в систему виключно як чорний текст. Потім система розпізнавання виділяє стовпці та рядки тексту, ділить відповідний текст на слова та масштабує зображення для зручності порівняння із "опорними" словами та

символами, вже встановленими на основі.

Крім того, розпізнавання тексту виконується символ за символом, використовуючи один із двох методів:

- метод "Порівняння із зображенням";

- метод "адаптивного розпізнавання".

Кожен із цих методів має певні переваги та недоліки та різні найкращі ситуації для їх застосування, які будуть розглянуті нижче.

Після того, як ми отримаємо отриманий результат після запуску програми, необхідним і дуже важливим кроком є збільшення швидкості програми та підвищення точності отриманих даних. Для цього необхідно додати певні словники, які допомагають системі з певного набору символів складати слова, які найкраще відповідають логіці цілого тексту, та виправляти деякі незначні неточності в словах. Дуже важливим способом підвищення точності вихідних даних є також використання системи розпізнавання з граматичними правилами, щоб правильно відрізнити іменник від дієслів та інших подібних випадків.

1.5. Алгоритми розпізнавання рукописних символів

При вирішенні проблеми розпізнавання друкованих і рукописних символів можуть бути задіяні певні структурні підходи, що дозволяють розділити зображення на окремі частинки, що становлять текст.

Спочатку ми розглянемо метод, який використовує топологічний опис зображень для цього завдання, які можуть бути представлені у вигляді плоских графів. При цьому в деяких окремих завданнях всі можливі зображення, складові певний клас, можуть бути виведені в результаті певних перетворень будь-якого зображення. Проблема розпізнавання в цьому випадку зводиться до встановлення збіжності представленого зображення з одним з порівнюваних. Його можна виявити за допомогою різних параметрів властивостей зображення, які не змінюються під час перетворень зображення. Інваріант, який дозволяє описувати зображення, наприклад положення певної точки, визначається кількістю рядків, що

сходяться в цій точці. Відповідний опис виходить перетином контурів зображення в певному порядку при фіксованому індексі точок. Встановлення гомеоморфізму - впізнавання самого по собі - зводиться до порівняння описів представленого зображення і еталонних зображень класів. Головне достоїнство цього опису - нечутливість до сильних змін зображення. Навчання топологічного коду полягає в створенні певного набору символів, які використовуються для порівняння зображень такого типу.

Інший подібний підхід - алгоритм розпізнавання подій. Цей підхід заснований на структурі об'єкта, яка визначається точками і лініями, і не змінюється при невеликих змінах зображень. Лінії, яке знаходиться під накладеної матрицею, визначають події. Образ події - це не тільки формальний набір функцій, але і адекватне топологічний опис. Вивчення підходу нейронної мережі зводиться до створення списку зображень купонів на досить великій кількості об'єктів обрізки. Переваги цього методу - відмовостійкість і висока продуктивність [11].

Недоліком таких методів розпізнавання є велика різноманітність і частота змін деформацій, які часто виникають на зображеннях з відсканованим текстом. Для класу чорнила розриви рядків - це передбачувані зміни. Пропонований алгоритм розпізнавання, який є структурним, поєднує в собі такі переваги, як висока швидкість і можливість оцінки відповідності між розпізнаними зразками і довідковими описами.

Зручний метод розпізнавання символів - це метод, заснований на подіях, і він має ряд недоліків, таких як відсутність результуючих значень, складність пошуку шаблонів, які перебувають не з одного пов'язаного компонента, а з декількох, а також велике поліпшення результату часто втрачається. спеціальні перетину, рівність шаблонів і т. д. d.

Відомий символ піддається проріджування (скелетирования). Є різні способи отримати скелети персонажів, які відрізняються один від одного. Подумайте про метод, запропонований Щепініним. Цей метод не входить в групу популярних в даний час алгоритмів паралельної скелетонізації, але не призначався для використання на машинах з паралельною архітектурою. У той же час у такого

підходу багато переваг. Так, цей метод зберігає 8-е з'єднання вихідного зображення. Це дозволяє дуже ефективно використовувати табличні дані, тому алгоритм працює на високій швидкості - понад 1200 символів в секунду на процесорі Core i3. Крім того, цей метод забезпечує подання скелета з пікселем, що виключає необхідність подальшої скелетизації зображення. Цей метод дозволяє зберегти всі особливості конкретного зображення, він зручний і дуже корисний при аналізі розпізнавання [11].

Спочатку вихідне зображення символу розділяється на компоненти зв'язку, для яких можна використовувати лінійне уявлення, сформоване для методу події. У кожному компоненті, для кожного зовнішнього і кожного внутрішнього шляху є початкові точки угорі ліворуч. Потім поетапно знімається шар точок. Для наступної точки контуру розглядається конфігурація восьми сусідніх точок. Точка видаляється, якщо вона не є кінцевою точкою (тобто не перебуває на початковому або останньому інтервалі прямою або поворотними лінії) і якщо вісім її сусідів після видалення все ще утворюють пов'язаний набір. Зверніть увагу, що з'єднання можна розуміти по-різному (8 з'єднань, 4 сполуки), тому ви можете легко отримати різні типи каркасних уявлень. Після аналізу точки і її сусідів і можливого видалення точки виконується перехід до наступної точки контуру так, щоб вона залишалася на краю зображення.

На одному етапі видалення шару точок для кожного компонента виконується по черзі для кожного зовнішнього, внутрішнього контуру. Процедура повторюється до тих пір, поки символ НЕ буде структурований однокрапкового лінією (рисунок 1.3.).

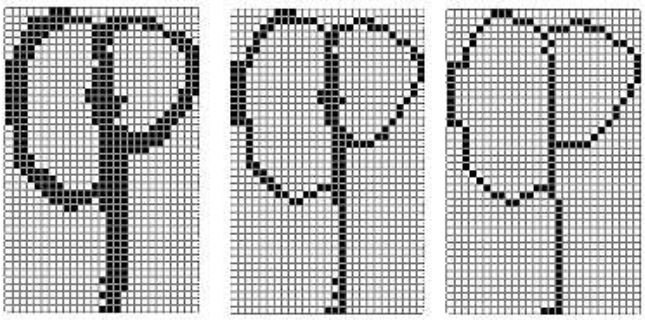

Рис.1.3. - Скелетізація образу, що складається з 1-го зовнішнього і 2-х внутрішніх контурів

Багато методи і підходи використовуються для підвищення швидкості рендеринга скелетного зображення. Наприклад, для отримання інформації про ймовірність точки або наступного переходу точки до граничного межі за допомогою раніше зіставлених таблиць, а не шляхом обчислення необхідних параметрів для піків.

Результат отримання символьного скелетного об'єкта не є останнім етапом розпізнавання пізніше, це структурний подання використовується пізніше для позначення основних властивостей символу. Основною причиною використання зображення скелетонізації і його подання є надана можливість отримати найбільш важливі функції головного героя, що містить текстові символи [11].

Скелетонізація діє як основна опорна сила в більшості систем ОРТ. Але для отримання правильного символу дуже важливий не сам метод скелетонізації (зазвичай скелети виходять дуже схожими, незважаючи на різні типи алгоритмів), а наступна привабливість уявлення, отриманого з алгоритму скелетонізації. Якщо ми завжди використовуємо для розпізнавання одні і ті ж функції плюс або мінус, в кінці ми часто отримуємо дуже схожі результати роботи.

Після цього перейдемо до теми залучення методу скелетонізації до завдань пошуку текстових символів. Після того, як зображення було отримано, воно піддається методам аналізу і реєстрації певних властивостей, наприклад, структури контуру або особливих характерних точок або коду ланцюга.

Особливі точки - останні точки і стартові сегменти - це точки, сусіди яких можуть утворювати не менше трьох пов'язаних областей. У прикладі на рис. 1.4 зображення має два внутрішніх контуру, кінцеву точку і 3 розділених сегмента або тріода. Кожен такий новий контур представлений у вигляді послідовного набору спеціальних точок і так званого ланцюгового коду, який може складатися з точки прив'язки, кількості кодів і вектора напрямку від попередньої точки до поточної точки.

В результуючому поданні відбувається повторний рендеринг, який складається з видалення коротких ліній, злиття близьких сегментів поділу і руйнування невеликих внутрішніх контурів.

Щоб правильно працювати із зовнішнім контуром, потрібно знайти його топологічний код або сам тип. Для цього необхідно представити контур як послідовний набір з числа особливих точок, що відповідають обраному симетричного напрямку за годинниковою стрілкою. Крім того, такі точки нумеруються, якщо змінюється і початок контуру, щоб спробувати визначити контур, з яким з основних типів. Статистичний аналіз освітніх віййбірці, що включає рукописні зображення заголовних букв українського алфавіту від «А» до «Я» і цифр від «0» до «9», показав, що контури містили типи, відмінні від найбільш важливих. менше одного відсотка [11].

Якщо ми вивчаємо інші можливі набори символів (малі літери, літери з іноземних алфавітів), будуть потрібні інші топологічні набори та коди. Тому для цієї мети була створена нова функція для побудови наборів топологічних кодів для даного набору. Досить просто створити і задіяти функцію для пошуку збігів тексту з використанням їх скелетного подання без обмеження кількості можливих топологічних кодів, що робить попередній аналіз можливих типів надлишковим, але непотрібним. Але необхідно підкреслити той факт, що використання обмеженого набору топологічних кодів також містить ряд переваг, так як дозволяє різним кодам легко застосовувати певні спеціальні функції, вибирати інші (додаткові) типи одиночних точок для певних кодів.

Ось деякі характеристики, пов'язані з проблемою ОРТ з використанням методу скелетонізації. Для кожної окремої точки в такому візуальному представленні віднімаються всі можливі символи:

- нормалізовані координати окремої точки (код графіка)

довжина ребра до найближчої точки пропорційна всій довжині графа;

- нормоване напрямок від заданої вершини до наступної особливій точці;

- нормалізований вектор для входу і виходу зверху;

кривизна дуги, точніше «ліва» і «права» криві дуги, які з'єднують вершину особливого числа з наступного точкою (кривизна вліво і вправо). Кривизну можна розрахувати як відношення максимальної відстані від точок дуги до прямої, що з'єднує вершини, до довжини відрізка, що з'єднує ті ж точки [11].

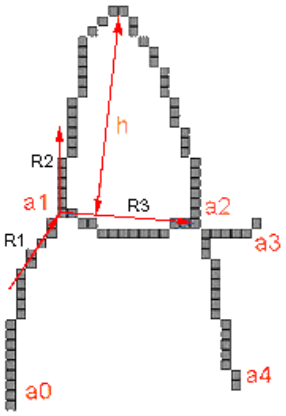

Рис.1.4. - Приклади скелетних ознак

На малюнку 1.4 показані найбільш важливі топологічні властивості літери «А». На кресленні чотири основних точки - a0, a1, a2, a3. Якщо обійти діаграму за маршрутом a0 ==> a1 ==> a2 ... У точці a1 відображаються наступні параметри: вектор r1 - вектор для входу в точку, вектор r2 - вектор виходу з точки, вектор r3 - глобальний вектор напрямку на наступну особливу точку ... Двохнаправлений вектор h відображає індикатор відхилення в ліву частину дуги (a1, a2) від прямої; «Правильне» відхилення - 0.

З наведеного опису видно, що кількість параметрів в 8 разів більше, ніж кількість кутових точок. Цей параметр відрізняється для кожного топологічного коду, і одні і ті ж параметри можуть мати різне значення для різних топологічних зображень.

Для деяких кодів кількість функціональних точок і кількість топологічних функцій дуже мало. Наприклад, для нульового символу взагалі немає топологічних знаків, тому що відсутня одна точка. Для інших топологічних кодів кількість властивостей може різко збільшуватися, вимагаючи достатньої кількості еталонних зображень для навчання мережі. Тому в більшості ситуацій в завдання ОРТ залучена лише певна частина властивостей.

Системне навчання складається з створення дерев розпізнавання для кожного заздалегідь визначеного топологічного коду.

Якщо конкретний символ після запуску методу перетину цього дерева залишається невизначеним, система намагається поліпшити зображення за допомогою наступних дій:

Якщо персонаж залишається невідомим після перетину дерева розпізнавання, спробуйте поліпшити зображення, виконавши такі дії:

- склейте кінці ліній за напрямками; Щоб вирішити таке завдання, необхідно обчислити вектори напрямків для всіх дуг в реалізації каркасного вирівнювання. У разі, якщо вектори напряму ліній схожі і вказують самі на себе, можна використовувати їх сполуки. Оскільки це може бути цілий рядок, вона була деформована під впливом певних помилок сканування або навіть набору тексту.

Обмежте точки склейки каркаса, які знаходяться на найкоротшій відстані один від одного;

- відхилення короткої лінії (графічної дуги) надмірно короткі строки зазвичай зустрічаються в почерку;

React не реагує на дрібні компоненти; якщо після аналізу з'ясувалося, що компонентів кілька, то, ймовірно, виникли додаткові перешкоди;

Прибрати внутрішній контур. У випадках, коли деякі з контурів досить малі, вони, ймовірно, утворюються після деякого ушкодження зображення або написи, а не ключових функцій символу.

Якщо зміни в скелеті неможливі, символ не ідентифіковано. В іншому випадку ми виконуємо модифікацію зображення за допомогою повторної скелетонізації, яка залежить від типу модифікації об'єкта. За кожну таку міру призначається штраф, який варіюється в залежності від параметрів операції або модифікації.

Серед наявних можливостей вибираємо набір операцій, який складається з елементів, що містять показник мінімального штрафу. Такий параметр розраховується за певною фіктивною точці, наприклад 256, яка задається певними символами. Потім отримані колекції можна об'єднати для подальшої роботи.

Характеристики методу розпізнавання символів зі скелетними функціями такі: точність - 95,6%, швидкість - 16 кадрів / хв. До переваг підходу також можна

віднести стійкість до певних деформацій, таким як обрив і вигин ліній, поява дрібних, ледь помітних ліній. Стійкість задається алгоритмом побудови вистави і дає методу перевага розпізнавання рукописних зображень, що характеризуються такими деформаціями [11].

Після всього вищесказаного можна сказати, що для даних ОРТ метод скелетонізації можна поліпшити за рахунок певних змін. Наприклад, значно попрацювавши на етапах навчання і визнання. Також можливо замість побудови топологічного дерева використовувати нейронну мережу, засновану на тих же топологічних властивості. Але також просте і легке використання першого методу і його продуктивність досить вигідні.

Також слід зазначити, що така програма, яка використовує метод скелетонізації, рідко використовується сама по собі, а тільки в поєднанні з деякими іншими методами розпізнавання, заснованими на зовсім інших принципах, в основному в поєднанні з нейронною мережею, заснованої на інших характеристиках. Таке поєднання різних методів розпізнавання рукописного введення дає досить хороші результати.

1.6. Розпізнавання рукописних символів

Проблема читання рукописних текстів стала досить серйозно обговорюватися ще в 1950-1960-х роках, одночасно з публікаціями по розпізнаванню образів. Одними з перших великих проектів були обробка пошти, анкетування і перепис. При цьому для реалізації процесу обробки використовувалося дуже дороге обладнання - оптичні зчитувальні пристрої. Згодом їх стали комбінувати з керуючими машинами, і в результаті система автоматичного зчитування перетворилася в комбінацію сканера і одного або декількох комп'ютерів. Сфера застосування цієї технології розширилася, збільшилася гнучкість, знизилася вартість, що призвело до можливості розвитку роботи і меншого масштабу, орієнтованого на бізнес-цілі.

Системи розпізнавання рукописного введення використовуються дуже по-різному: створення електронних бібліотек і архівів шляхом перетворення книг та

інших документів в цифровий комп'ютерний формат; можливість переходу з паперового на електронний документообіг компаній; твердження квитанцій про узуфрукт та інших платежах; прийом страхових випадків; оформлення авто при реєстрації, оформлення документів про ДТП та інше.

Існує два типи технології розпізнавання рукописів: автономна (off-line) - традиційна технологія читання паперових документів і пізніше - «оперативна» (онлайн), в останньому використовується перо і спеціальний екран як інструмент, чутливий до рух пера і фіксація dess bana. У свою чергу, офлайн-розпізнавання ділиться на кілька підвидів, для кожного також існують окремі методики. Таких підвидів всього три: розпізнавання рукописного тексту, рукописне, просто друковане. Проблема розпізнавання рукописного введення є найскладнішою з трьох, і існуючі в даний час технології в цій галузі можуть використовуватися тільки для вирішення дуже обмежених завдань, таких як розпізнавання прийому. Як і у випадку з розпізнаванням об'єктів, основна проблема полягає в тому, що кількість можливих написань будь-якого символу або їх комбінацій можна назвати нескінченним. Тому їх класифікація не тільки складна, але і дуже складна.

Якість розпізнавання досить сильно залежить від якості фотографій із зображенням тексту, воно визначається формою персонажів, стилем письма та виконанням. Знак, який легко впізнати за формою, є ознакою хорошого стилю, так як він не містить зайвих елементів. На практиці вимоги до якості символів не завжди виконуються, тому прямі помилки і конфліктні ситуації усуваються за допомогою логічних перевірок і перевірки словника і втручання оператора [12].

Незважаючи на те, що багато наукових колективів досягли значних результатів у вирішенні проблеми розпізнавання тексту, на сьогоднішній день залишається невирішеним велику кількість проблем, вирішення яких може підвищити ефективність розпізнавання нестандартних символів автоматичними системами, в результаті з яких для підвищення ефективності процесу введення та обробки текстової інформації ... зокрема, найбільш важливими завданнями є:

- оцінювати ефективність систем розпізнавання символів,

- оцінити оптимальний розподіл процесу розпізнавання в апаратному та

програмному забезпеченні;

- підвищена инвариантность за масштабом і уточнені перетворення;
- сегментація рукописного тексту на окремі символи;
- завдання виділення інформативних функцій нестандартними символами;
- розробка ефективних методів класифікації графічних зображень символів.

Висновки за розділом

У цьому розділі розглянуто актуальність і історія розвитку розпізнавання тексту. Крім того, було проведено аналіз існуючих методів розпізнавання символів, на підставі якого було виявлено ряд недоліків і переваг кожного методу. Концептуальний апарат також був представлений для того, щоб можна було повністю орієнтуватися у всіх розглянутих в роботі методах.

РОЗДІЛ 2
РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ НА ОСНОВІ ПЕРЕДБАЧЕННЯ

2.1. Проблеми створення системи

Окуло-рухова активність - важливий компонент поведінки людини і тварин, що підкреслює велику частину повсякденних рухів і взаємодій з об'єктами, пристроями або людьми. Переміщаючи погляд очима, центр зору постійно і активно переміщається. Сканування візуального образу відбувається в основному з використанням високошвидкісних рухів очей і називається Сакада (Ярбус (1967)) і послідовно охоплює певні частини візуального образу. Незважаючи на те, що цей процес часто зустрічається в природі, розпізнавання предметів Сакадою рідко розглядається в штучному зорі. Є багато причин, в тому числі високопродуктивні датчики, що забезпечують мільйони пікселів за невисокою ціною. Потім розробляються все більш і більш потужні комп'ютерні пристрої, які одночасно обчислюють мільйони пікселів для розпізнавання і споживають ресурси грубої сили [14].

Однак приклад зору тварин вимагає пошуку нових методів використання більш простих алгоритмів. Основним аспектом зору тварин є використання активних скануючих пристроїв, які можуть переміщатися з певним ступенем свободи для вибору певної точки огляду. Наявність набору можливих сенсорних рухів вимагає розробки спеціальних алгоритмів, які повинні вирішувати проблему вибору точки зору.

Активна програма комп'ютерного зору повинна, наприклад, проаналізувати попередній досвід, щоб зрозуміти, якою має бути точка зору.

Використовуйте для надання найбільш корисної інформації про візуальному зображенні.

Оптимізація процесу для періодично переміщаються датчиків може бути частиною алгоритму комп'ютерного зору в поєднанні з традиційними пиксельними операціями.

Загалом, ідея вибору уявлень повинна в першу чергу враховувати розрахунки, які необхідно зробити для досягнення конкретного завдання. Наприклад, віртуальне скануючий пристрій буде діяти як фільтр, який буде вибирати, яка частина сигналу буде досліджена, а яка частина буде пропущена. Це можна використовувати для роботів і дронів, яким потрібно швидкий відгук за допомогою легких і слабких пристроїв сканування. Точно так же комп'ютерний зір вимагає мегапіксельних зображень з високою роздільною здатністю для виборчого стиснення зображень, щоб уникнути непотрібних результатів матриці. Менш інтуїтивно постійно зростаючі бази даних для студентів і машинного навчання також забезпечують інтелектуальний пошук даних, так що тільки важливі приклади або властивості, в залежності від контексту, зберігаються до того, як вони будуть навчені ім. Таким чином, крім проблеми вибору точки зору, існує проблема вибору характеристик, які повинні ґрунтуватися на контексті.

Концепція активного бачення і / або активного сприйняття зустрічається в літературі з робототехніки з різними твердженнями. У Aloimonos et al. (1988) автори розглядають ВІАД для обробки зображень з декількох точок зору, тобто показують, що деякі проблеми некоректного розпізнавання об'єктів стають правильними, коли один і той же об'єкт розглядається з кількох точок зору. Цей термін також був запропонований Баджо (1988) в якості дорожньої карти для розробки систем штучного зору, які забезпечують першу інтерпретацію активного зору в зв'язку з байєсівської послідовної оцінкою.

Парадигма активного утримання була введена незалежно в неврології. Загальний параметр, запропонований Фрістоном і його колегами, полягає в тому, що загальна тенденція мозку долати непередбачувані сенсорні події шляхом побудови генеративних патернів з часом покращує його прогнози і спрощує

сприйняття світу. Це поліпшення в основному досягається шляхом створення вибірок з середовища і видалення статистичних інваріантів, які використовуються для прогнозування майбутніх подій. Таким чином, побудова такої моделі засновано на витяганні певного набору інваріантів і їх організації для ефективної обробки надходять сенсорних даних за допомогою прогнозує кодування (див. Rao and Ballard (1999)). Ця пропозиція схоже на теорію автокодування, запропоновану Хінтон і Лендса (1994), але вводить новий погляд на його кодування, формально пов'язуючи побудова зображення даних і (оптимальне) управління двигуном. Зокрема, моторний контроль розглядається як конкретна реалізація процесу відбору, тобто як основа для оцінки складного апостеріорного розподілу.

2.2. Нейронна мережа в задачах обробки зображень

У складних завданнях від ОРТ ситуація вимагає створення систем на основі нейронних мереж. Вони можуть містити властивості, орієнтовані на рішення конкретних завдань, з урахуванням таких властивостей об'єкта, як: збільшення, орієнтація в просторі, візуальні індикатори [15]. Тим часом, нейронні мережі мають такий істотний недолік, як відсутність певних кваліфікованих і ефективних методів, що вирішують проблему пошуку відповідних об'єктів для динамічних зображень. Основні проблеми при виконанні цього завдання - великий розкид функцій і певна деформація об'єктів в просторі. Стиснення просторових властивостей здійснюється шляхом пошуку інтегрованих і незмінних змін візуальних перетворень креслень. Важливим засобом для вирішення задачі розпізнавання образів є метод алгебраїчних перетворень і геометричних інваріантів. Так, наприклад, інваріанти правильно використовуються для виявлення об'єктів і резервуарів повітряного простору, рукописних і друкованих букв, властивостей стикувального вузла космічного корабля і багатьох інших подібних об'єктів. На основі таких властивостей в моїх фотографіях заснована теорія алгебраїчних перетворень.

Нейронні мережі складаються з елементів, званих формальними нейронами. У цьому випадку кожен з нейронів отримує безліч вхідних сигналів, які надходять в його канали від групи дуже схожих нейронів. Після цього відбувається обробка даних, які вже містять попередні сигнали та адаптуються до них за допомогою процедур навчання. Потім результати відправляються наступного набору нейронів. Індикатор зв'язку між нейронами має ваги, які показують, наскільки важлива їхня інформація для нашої мережі і для отримання результату. Основним підходом до налаштування ЯМ є використання методів оптимізації та змін, що вносяться виходячи з певних характеристик, можливість перенавчання. Найбільшою перевагою NM є те, що всі індикатори можуть працювати паралельно. Через це результат розпізнавання збільшується, особливо при роботі з даними, які надходять в реальному часі. Системи вирішення проблем ОРТ, засновані на ЯМ, використовують ієрархічну структуру. Спочатку напрямок властивостей змінюється під впливом високого рівня помилки, але пізніше система вчиться все більше і більше, щоб знаходити результати за набагато коротший час.

Більшість цих ЯМ вимагають використання багатошарових перцептронів. Своєю популярністю вони зобов'язані досить широкому колу можливих проблем, які вони можуть вирішити. Найчастіше вони вирішують завдання з апроксимацією багатовимірних функцій, тобто будують багатовимірне відображення F: x → y, узагальнюють певну кількість пар даних (еталонних відображень) {xa, ya}.

Залежно від експоненти вихідних змінних (вхідні значення не розігруються вирішальним чином) апроксимація функції може бути представлена як

- класифікація (дискретний набір початкових значень);

регрес (безперервне виробництво).

Всі основні завдання розпізнавання об'єктів (зображення, фільтрація шуму, робота з тимчасовими рядами) можна звести до наступних основних завдань.

Подумайте, як навчити його виконувати звичайну задачу. Процес навчання перцептрона здійснюється шляхом надання колекції (набору) фотографій, по

одному запису за раз, і зміни ваг до тих пір, поки всі зображення не знайдуть оптимальний результат. Наприклад, вхідні фотографії намальовані на яскравих картах. Всі картки розділені на квадратні зони, і з кожної такої зони на глядача відправляється вхідний сигнал. Якщо зона містить лінію, від неї виходить 1, інакше - 0. Набір таких зон на карті визначає набір 0 і 1, які відправляються на входи перцептрону. Основна мета - вивчити таку систему, щоб включити покажчик при застосуванні набору вхідних даних, що відображають непарне число, і не включати в разі параду. Малюнок 2.1 відображає цю конфігурацію.

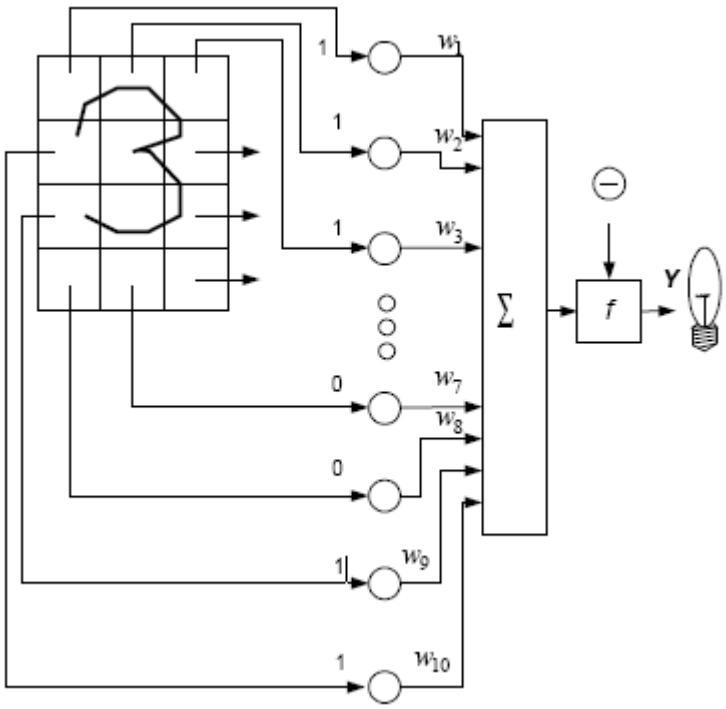

Рис. 2.1. – Система розпізнавання образів у перцептроні

Ми можемо оцінити, що напрямок x відображає демонстраційну карту, встановлену для розпізнавання. Кожен компонент xi в напрямку x множиться на його протилежний компонент wi на напрям w ваг. Тоді ці результати такі: в разі, коли сума більше порогового значення q, вихід нейрона y дорівнює 1 (покажчик включений), в інших ситуаціях - 0. Цей ефект можна перевести у векторну форму y = xw. Для вивчення НМ на вхід відправляються відображення x і обчислюється початкове значення y. Коли є правильний вихід, нічого не відбувається. Але коли

результат не відповідає дійсності, до вхідних даних додається ваговий коефіцієнт, щоб зменшити помилку.

Подивимося, як це робиться. Будемо вважати, що така картка із зображенням числа 3, представлена на вході і виході у, дорівнює одиниці (NN показує нам непарне). Ось так ми отримали правильний результат, це не поправляє ваги. Коли карта номер 4 надходить на вхід, а вихід у дорівнює 1 (непарне число), ваги, прикріплені до окремих входів, повинні бути зменшені, оскільки вони дають нам невірні дані. Точно так же, якщо карта номер 3 забезпечує 0 вихідних даних, ваги, прикріплені до перших входів, повинні бути збільшені, щоб виправити розмір помилки.

Для обмеженої кількості прогонів NM, щоб знати, як правильно розділити карти за принципом парності, за умови, що числа розділені лінійно. Для всіх непарних карт висновок буде більше певного порогу, а для всіх парних карт - нижче. Варто відзначити той факт, що навчання відбувається глобально - НМ можна навчати за допомогою всього набору вхідних показників.

Персептрони зазвичай популярні, тому що вони вирішують проблеми в своєму напрямку і є універсальним і ефективним способом з точки зору обчислювальної складності пристроїв.

Недоліком такого підходу є перевантаження НМ в ситуації, коли ми необгрунтовано збільшуємо кількість нейронів. Ще один недолік полягає в тому, що існує так багато функцій, які неможливо сегментувати за допомогою одношарової нейронної мережі. Такі функції називаються лінійно нерозривними, тому ми змушені накладати обмеження на використання одношарових ЯМ.

2.3. Основні принципи роботи алгоритму

Активний висновок заснований на довгій історії імовірнісного моделювання в обробці сигналів і управлінні. Формально фізичний світ набуває форми породжує процесу, який викликає сенсорний потік. Цей процес не бачимо сам по собі, але відчувається тільки через процес вимірювання шуму, який дає вектор

спостереження x. Завдання виведення полягає в тому, щоб оцінити основні причини спостереження на основі прихованого вектора z і управління u. u з поточної оцінки z, щоб максимізувати точність процесу оцінки. Це мета контролера сприйняття [16, 17, 18].

Замість того, щоб вибирати u випадковим чином, загальна мета активної кінцевої структури полягає в тому, щоб вибрати u таким чином, щоб мінімізувати поточну невизначеність щодо z в кращому випадку. Знання z може бути відображено в зворотному розподілі ρ (z). Чим краще знання (точність) сенсорної сцени, тим нижче ентропія, s: $\mathrm{H(p)} = \mathrm{E_{z\sim p}}[-\log \mathrm{p(z)}]$

Мінімізація апостеріорного значення ентропії дією може бути пов'язана з мінімізацією змінної змінної енергії, пов'язаної з сенсорною сферою [19]. Таким чином, очікується, що управління буде зменшувати ентропію r на кожному етапі. Це оптимальне значення u невідомо заздалегідь, тому що x визначається тільки після вирішення цього рівняння. Потім визначається структура, яка визначає ефект і його найбільш ймовірні результати відповідно до генеративної моделлю.

Повертаючись до попереднього кроку, загальна формула генеративної моделі являє собою основу для управління зі зворотним зв'язком в формалізмі дискретного байєсівського виведення. З огляду на початковий стан z0, прогнози відновлюють постійні розподілу, а саме динаміку P (Z | u, z0) зв'язків, що генерує процес вимірювання Z - і P (X | z, u), що генерує x -. Потім для майбутнього тилового розподілу (правило Байеса):

$$\mathrm{P(Z|X,u,z_0)} = \frac{\mathrm{P(X,Z|u,z_0)}}{\mathrm{P(X|u,z_0)}} = \frac{\mathrm{P(X|Z,u)P(Z|u,z_0)}}{\sum_{z'} \mathrm{P(X|z',u)P(z'|u,z_0)}}$$

майбутня ентропія: $\mathrm{E_X[H(p)|X,u,z_0]} = \mathrm{E_X[E_Z[-\log P\,(Z|X,u,z_0)]]}$

і оптимальне U є: $\tilde{u} = \underset{u\in U}{\operatorname{argmin}}\, \mathrm{E_X[H(p)|X,u,z_0]}$

Аналітичні розрахунки недоступні (для прогнозування наступного розподілу x). Розглянути оцінку ~ u, яка повинна спиратися на відбір проб від генеративного процесу, щоб передбачити ефект u, тобто: $\tilde{u} = \underset{u}{\operatorname{argmin}} \frac{1}{N} \sum_{\substack{i=1..N \\ x^{(i)}\sim P(X)|u,z_0) \\ z^{(i)}\sim P(Z|x^{(i)},u,z_{0)}}} -$

$\log P(z^{(i)}|x^{(i)},u,z_{0)}$

або навіть більш чітка пряма оцінка за допомогою оцінок максимальної правдоподібності (точкова оцінка):

$$\tilde{x}_u = \arg\max_x P(x|u, z_0)$$

$$\tilde{z}_u = \arg\max_z P(z|\tilde{x}_u u, z_0)$$

$$\tilde{u} = \arg\min_u -\log P[\tilde{z}_u|\tilde{x}_u, u, z_0]$$

Цю операцію можна повторювати послідовно, де за фактичної модуляцією u = u ~ слід зчитування фактичного спостереження x, що, в свою чергу, дозволяє оновити фактичне апостеріорне розподіл по z. Цей оновлений A posteriori стає апріорі для наступного кроку рішення, тобто z1 ~ P (Z | x, u, z0), тому може бути здійснена нова модуляція u1 і т. Д.

Якщо позначити T як останній крок в процесі, де u0: Ti - фактична послідовність управління, а x1: T - фактична послідовність спостереження; остання апостериорная оцінка - це P (Z1: T | x1: T, u0: T 1, z0), що відповідає оцінці POMDP (частково спостерігається марковский процес прийняття рішень) (див. рисунок 2.2), суть якого буде визначатися принципи мінімізації ентропії, певні вище, саме для полегшення процесу оцінки. Таким чином, активна система логічного висновку виявляється сприйнятливою політикою (основна мета - полегшити процес оцінки).

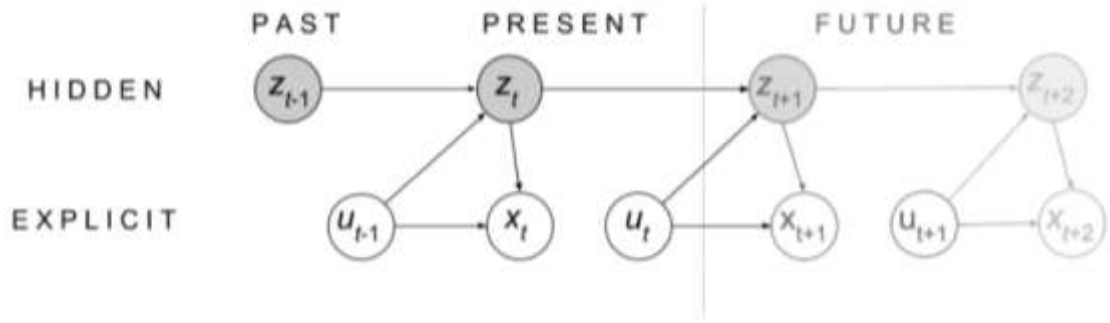

Рис.2.2. - Генеративна модель

Логіка активного зору полягає в зовнішній візуальної сцені X, яка ніколи повністю не розкривається, ви можете розглядати тільки певну частину сцени x під час орієнтації датчика (як це відбувається в режимі кращого відображення). Знаючи, що z не змінюється при зміні положення u-датчика, виявлення z має

полягати в зборі ділянок x датчика за допомогою зміни u (орієнтації датчиків) з плином часу для уточнення оцінки z. ρ0 (z), вибір u супроводжується переглядом візуального образу, дає x, що, в свою чергу, дозволяє уточнити оцінку z. Кожна саккадой повинна консолідувати припущення про затримку z, яка може зберігатися і поширюватися крок за кроком до тих пір, поки не буде зібрано достатню кількість свідчень [19].

Структура активного бачення допускає безліч різних спрощень у порівнянні із загальною схемою оцінки POMDP, враховуючи, що зміна u не впливає на компоненти стадії, тобто P (Zt + 1 | u, zt) = P (Zt + 1 | zt). Потім зі статичним допущенням передбачається, що не повинно бути ніяких значних змін в сцені під час процесу дослідження мішка, тобто ∀t, t ', zt = zt' = z. Це, нарешті, означає спрощену ланцюжок оцінки після hoc:

$$P(Z|x_{1:t+1}, u_{0:t}, z_0) = \frac{P(x_{t+1}|Z, u_t)P(Z|x_{1:t}, u_{0:t-1}, z_0)}{\sum_{z`} P(x_{t+1}|z`, u_t)P(z`|x_{1:t}, u_{0:t-1}, z_0)}$$

випливаючи в остаточну оцінку P (Z | $x_{1:T}$, $u_{0:T-1}$, z_0).

Інтерпретація: структури активного виведення, пов'язані з теорією автокодування (мінімізація вільної енергії) і прогнозуючим кодуванням, забезпечують чітку дорожню карту для ефективної реалізації в штучних одиницях. Він повинен спиратися на три елементи, а саме:

- генеративна модель p, яка передбачає наступний вигляд x під поточною думкою z0 та точка зору u, $p(.|u, z_0) \cong \sum_{z`} P(X|z`, u)P(z` 0 |u, z_0)$

- модель висновку q, яка передбачає наступний z під видом $\tilde{x}$ та точкою зору u, тобто: $q(.|\tilde{x}, u, z_0) \cong P(Z|\tilde{x}, u, z_0)$

(Використовувати динаміку зв'язків P (Z | u, z0), неявно вбудовану як в генеративні моделі, так і в моделі результатів в цілому) - і політику π, яка повинна використовувати двоетапний прогноз (спочатку см. Попередні прогнози, а потім висновок в заданій формі) дати оптимальне управління згідно з усіма рівнянь (5) або рівнянням (6-8).

З комп'ютерної точки зору і по відношенню до z = z0 (припущення про статичної сцені) кожна різниця u відповідає різної точки зору для статичного

зображення з набором породжують {pu (. | Z)} u∈U і висновки {qu (. | X) } u ∈U моделі систематично вивчаються для кожного виду u. Ці слабкі локальні класифікатори контрастують з низькорівневими локально інваріантними фільтрами, використовуваними в традиційній візуалізації [20], і / або з першим рівнем згортальних фільтрів, використовуваних в сверточное нейронної мережі.

2.4. Реалізація системи розпізнавання тексту на основі динамічного передбачення

Як попереднього кроку ми вважаємо, що гіпотетична і генеративних моделі вивчаються окремо, щоб мати можливість окремо оцінювати характеристики стратегії управління. Цей заснований на моделі підхід до послідовного вибору ракурсів представлений в алгоритмах 1 і 2.

Важливим алгоритмическим терміном в порівнянні з формулами (6-8) є використання набору динамічних дій: U. На кожній ітерації нове вбрання дію u ~ витягується з U, так що наступний вибір буде проводитися в нових напрямках. ще не вивчено. Це реалізує принцип придушення зворотного зв'язку [21].

- Другий алгоритмічний аспект - це використання порога Href, щоб зупинити накопичення доказів, коли буде зібрано достатньо доказів. Цей поріг є вільним параметром для алгоритму, який вказує, надаємо ми привілеї консервативному (щільному) або оптимістичним (порожньому) порогу. Критерій завершення повинен бути оптимізований, щоб розрізняти збереження ресурсів і точність кодування.

МОДЕЛЬ НА ОСНОВІ FOVEA (базова модель)

Більшість хребетних використовують два основних прийоми, щоб звести до мінімуму споживання сенсорних ресурсів на етапі дослідження. Перший трюк - це ямчатая сітківка, яка концентрує фоторецептори в центрі сітківки з більш недостатнім розподілом по периферії. Ямчатая сітківка дозволяє обробляти центральні просторові частоти і периферійні низькі частоти одночасно (т. Е багатомасштабного обробка паралельно). Другий трюк - це вже згадане послідовне

вивчення саккадических сцен, яке дозволяє фіксувати інформацію про високу частоту простору там, де це необхідно (послідовна обробка) [19].

Алгоритм 1. Політика на основі прогнозів.

Елементи: p (генератор), q (висновок), ρ (апріорі), U (набір дій)

ge z ~ ρ

$\forall_u \in U$, генеруємо $\tilde{\boldsymbol{x}}_{\boldsymbol{u}} \sim u \sim p(x \mid z, u)$

Повертаємося до $\tilde{u} = \underset{u \in U}{\operatorname{argmax}}\, q(z|\tilde{x}_u, u)$

Алгоритм 2. Вивчення сцени.

Необхідні елементи: p (генератор), q (висновок), ρ0 (початковий апріор), U (задані дії)

ρ ← ρ_0

тим часом, коли H (ρ)> H_{ref}, виконуємо

обираємо: $\tilde{\boldsymbol{u}} \leftarrow$ Політика, що базується на передбаченні (p, q, ρ, U)

зчитуємо: $\boldsymbol{x}_{\tilde{\boldsymbol{u}}}$

оновлення: $\forall_z$, непарний [z] ← log q (z | $\boldsymbol{x}_{\tilde{\boldsymbol{u}}}$, $\tilde{\boldsymbol{u}}$) + log ρ (z)

ρ → softmax (odd) (апостеріор стає апріором наступної черги)

U ← U \ {$\tilde{\boldsymbol{u}}$)}

тим часом повертаємося до ρ

Пропонована базова модель бачення спочатку призначена для вивчення місцевих бажаних видів зображення. Візуальне перетворення обмежена його основними алгоритмічними елементами, тобто локальним стисненням зображення по певної спрямованості [22]. Таким чином, стиснення фовеальній зображень засновано на «піраміді» двовимірних хвильових коефіцієнтів Хаара, поміщених в поле зору. Використовуючи базу даних MNIST як приклад, ми спочатку перетворимо вихідні зображення відповідно до 5-рівневим вейвлет-розкладанням (див. Малюнок 2.3-b). Потім ми визначаємо точку огляду u як набір трьох координат (i, j, h) з індексом в рядку, індексом стовпця j і просторовим масштабом h. Кожному u може відповідати поле зору, що складається з трьох хвильових коефіцієнтів xi, j, h ∈ R15, отриманих за допомогою горизонтального,

вертикального і похилого фільтрів на місці (i, j) і шкали h. Мультімасштабная візуальна інформація xi, j ∈ R 15 знаходиться в координатах (i, j), відповідає набору з 5 коефіцієнтів триптиха, а саме xi, j = (xi, j, 5, x [i / 2], [j / 2] 4, x [i / 4], [j / 4], 3, x [i / 8], [j / 8], 2, x [i / 16], [j / 16], 1} (див. рис. 2.3-c), так що кожне багатозначне поле зору має 15 коефіцієнтів (і на відміну від 784 пікселів в початковому вигляді). На рис. 2.3-b показано відтворене зображення з 4 центральних точок огляду в координатах (7, 7), (7, 8) (8, 7) і (8, 8).

Для кожної моделі u = (i, j, h) (всього 266 слабких моделей) вивчено понад 55 000 прикладів з бази даних MNIST. Для кожної категорії z і кожної орієнтації точки зору генеративная модель будується з набору параметрів Θz, u = (ρz, u, μz, u, Σz, u), так що ∀z, u, ⟦x _⟧ _U, u ~ B (ρz, u) × N (μz, u, Σz, u) з B розподілом Бернуллі і N різноманітним гауссовским. Бернуллі повідомляє про випадки, коли триплет коефіцієнтів дорівнює нулю в переглядається частини зображення (це досить часто на периферії зображення), що призводить до того, що відповідний триплет відкидається при обчисленні гауссових моментів. Кожна результуюча слабка генеративная модель p (X | z, u) являє собою суміш Гаусса / Бернуллі для 10 міток MNIST. Для моделі укладення зворотна сторона може бути явно розрахована тут за допомогою правила Байеса, тобто q (Z | x, u) = softmax log p (x | Z, u).

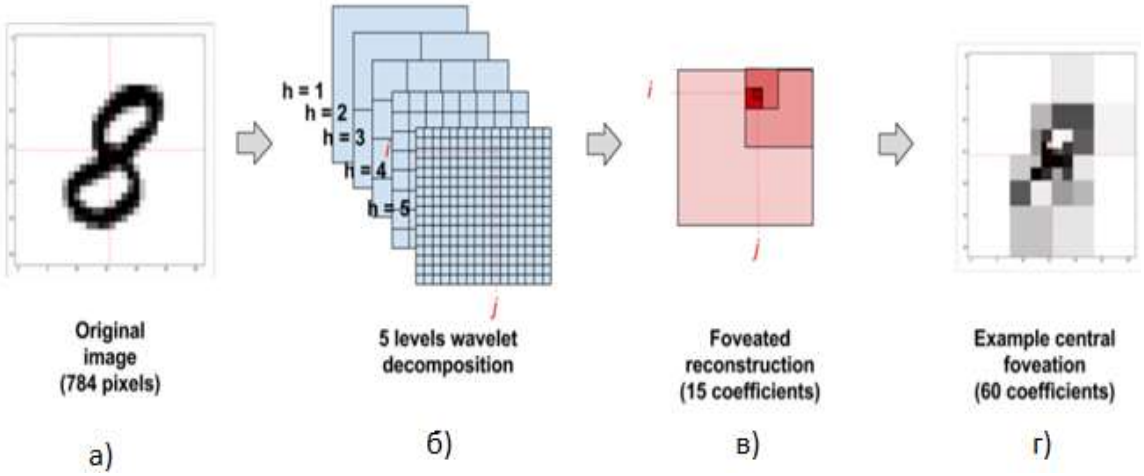

Рис.2.3 - Побудова зображення з фовеатом

Алгоритм вивчення саккад є адаптацією алгоритму 2. Процес починається з довільного припущення, яке ґрунтується на інтерпретації кореня хвильового

коефіцієнта зображення, з якого формується вихідне припущення ρ0. Потім кожен наступний мішок розраховується на основі остаточних координат (i, j) ∈ [0 .. 15] 2, так що зворотний розрахунок заснований на декількох потрійних коефіцієнтах. Коли вибирається (i, j), всі відповідні координати (h, i, j) відкидаються з U і не можуть бути повторно використані для майбутньої зворотного оцінки (для остаточної зворотного оцінки її можна узгодити з рівномірним скануванням по хвильовим коефіцієнтами).

Приклад такого дослідження зображення мішка показаний на малюнку 2.4-a для більш ніж одного зразка MNIST. Стан процесу розпізнавання після провалу показано на малюнку 2.4-b. Наступний мішок (рис. 2.4-c) веде в область зображення, яка, як очікується, допоможе підтвердити припущення. Триваюча саккадой (рис. 2.4-d) виконує обстеження, а остання саккадой (рис. 2.4-d) дозволяє вам досягти порога апостеріорної ентропії, встановленого тут Href = 1e-4. У другому рядку показано накопичення свідоцтв перед коефіцієнтами трійок, і на рис. 2.4-ea posteriori показано оновлення різних написів, а на рис. 2.4-e показана апостериорная ентропія поновлення відповідно до коефіцієнтів трійок. читати. Важливо, щоб для кожної позиції кінцевого ефектора (i, j) зчитувалося кілька кодонів (див. Рис. 2.3-c). Наприклад, у вихідній орієнтації очі (5) зчитували 5 трійок, а потім 4 трійні для кожного продовження мішечка.

Модель показує візуально реалістичні мішки, оскільки вони охоплюють весь діапазон зображень і мають тенденцію фокусуватися на областях, що містять клас пікселів. Реконструкція зображення після чотирьох саккад дозволяє візуально розпізнати «нечіткі» три, хоча в цьому немає необхідності, якщо саккади були обрані випадковим чином. Отриманий шлях ілюструє логіку вгадування підтвердження, яке знаходиться за межами активної точки зору. Кожен мішок спрямований на інтервал, який підтвердить поточну гіпотезу. Це твердження на перший погляд здається суперечливим, оскільки деякі очікують, що погляд буде спрямований на області, які можуть спростувати це припущення (заперечують поточну гіпотезу). Фактично, це не той випадок, коли інтервали, що підтверджують клас, рідше, ніж інтервали, що відхиляють клас, тому перехід до інтервалу, що

підтверджує клас, може надати більше інформації, якщо вихідне припущення стає несподівано невірним.

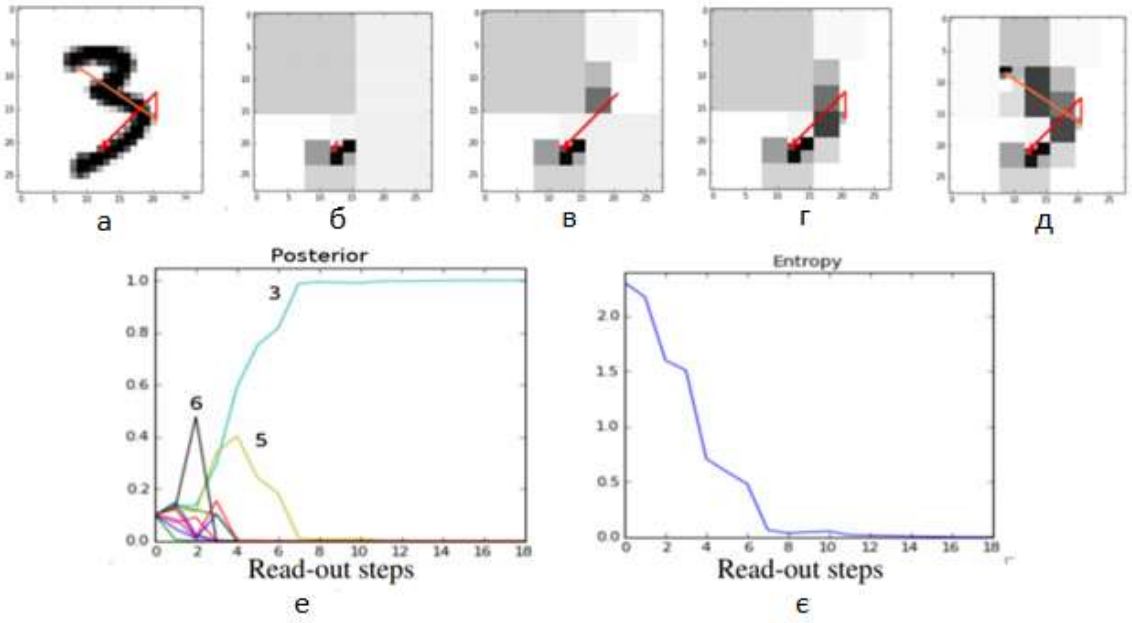

Рис.2.4. Вивчення сцени через саккади в моделі foveated vision.

а) Траєкторії мішковини по вихідного зображення (вихідна орієнтація точки огляду позначена червоним "плюсом"). be - прогресивна реконструкція зображення з мішками, з коефіцієнтами: б) 5 коефіцієнтів трійок + кореневої коефіцієнт (вихідна орієнтація точки зору), в) 9 коефіцієнтів трійок + кореневої коефіцієнт (перші мішки), г) 13 коефіцієнти для триплетів + кореневої коефіцієнт (другі пакети) е) 17 триплетних коефіцієнтів + кореневої коефіцієнт (треті саккади) f) апостеріорне оновлення в залежності від кількості кроків читання (зверніть увагу, що крок 1 залежить від кореневого коефіцієнта і наступних кроків для читання 3 Вейвлет-коефіцієнти Хаара), з одним кольором для кожної категорії (кількість рядків, представлених конкуруючими наклейками): Задня ентропія оновлюється в залежності від кількості кроків читання.

Значення політики (політика спільного використання на піксель)

При перегляді великих зображень складно масштабувати модель. Політика вводиться в два етапи прогнозування (рівняння (2.6-2.8) і алгоритм 1), які аналогічні значенням O (| U |. | Z |), оскільки він прогнозує наступне зворотне розподіл по z для кожного візуального елемента xu прогноз. Для порівняння:

параметрезованих політика більш ефективна для обчислень, що означає, що операція може виконуватися над заходами в заданому контексті. На щастя, таку параметризрвані політику тут легко обчислити. Беручи z0 в якості першого припущення і відзначаючи візуальне генеративное пророкування x _ (u, z_0), коли z0 перекладається в візуальну орієнтацію u, за умови, що рівновага апріорі по затримок, незалежно від процесу апостеріорного дослідження, так само

$$p_{u,z_0}(.) = \frac{p(\tilde{x}_{u,z_0}|.,u)}{\sum_{z`} p(\tilde{x}_{u,z_0}|z`,u)}$$

забезпечити для кожного (u, z0) автономний прогноз, а саме ρu, z0 (z0). Цей автономний розрахунок дає для кожного припущення z0 вашу попіксельно карту.

Карти сегментації на піксель на низькому рівні взяті з [21], з безліччю спостережень і поліпшень стиснення зображення / відео. У нашому випадку ця карта обробляється для кожного припущення z0, що призводить до вибору точки зору щодо підтвердження z0. Важливість політиків дозволяє визначити оптимальний шлях провисання способу, який слід послідовності «основних» поглядів на спадаючу важливість (відповідно до забороною повернення). У нашому випадку точка огляду, обрана на кроці t, залежить від поточного припущення zt, з перемикачем карти в русі, якщо припущення змінено під час мішка.

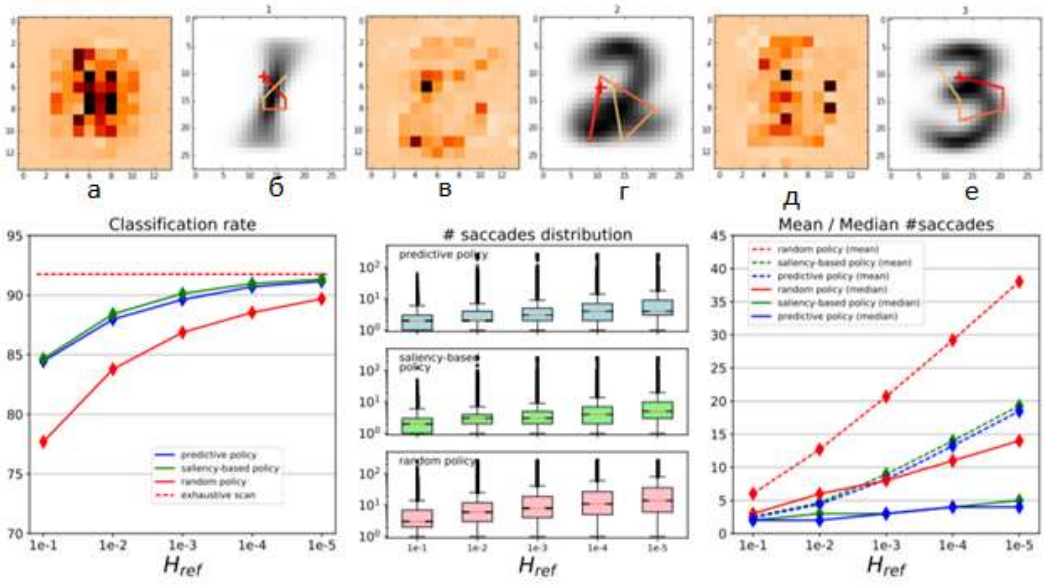

Рис.2.5. Політика значимості- верхня панель: карти попіксельного сегментування, виведені з моделі, з відповідними прототипами траєкторій саккад.

а) Прихована піксельна карта сегмента для прихованого класу «1». б) саккадических шлях Прототипу 5 для прихованого класу «1» (початкова позиція відзначена червоним «плюсом») для середнього класу. в) Карта сегментації на піксель для прихованого класу «2». г) Прототип 5 саккадических орбіти для латентного класу «2» в середньому класі. д) Карта сегмента на піксель для прихованого класу «3». (F) Прототип 5-саккаденовой орбіти для латентного класу «3» в середньому класі. Нижня панель: порівняння політик (зліва). Класифікація для передбачуваною політики, політики попиксельного підвищення різкості і випадкової однорідної політики для різних порогових значень. Швидкість вичерпного розпізнавання (базова) визначається в швидкому темпі. (Середнє) Кількість секцій мішка для політики прогнозування, політики попиксельного підвищення різкості і політики випадкового вибору. Площі вказують на першу і третю чверті. (Праворуч) Середнє і медіанне кількість мішків в залежності від порога розпізнавання для різних розглянутих політик.

Приклади таких карт сегментації на піксель показані на верхній панелі малюнка 4 для категорій з 1 по 3. Ці карти дозволяють детально аналізувати точне місце розташування класу (який виглядає коричневим), протилежної невизначеним класах сайтів (від блідо-оранжевого до білого). Перше, на що слід звернути увагу, - це відносна відсутність конкретних класів. Ці сайти, які «надають докази» здаються, як і очікувалося, взаємовиключними від класу до класу. Очікується, що невелика кількість мішків надасть більшу частину інформації про класифікацію, в той час як інша частина зображення буде неінформативної (або навіть неінформативної, якщо буде білуватий колір). Другий аспект полягає в тому, що всі відповідні місця розташовані в центральній частині зображень, тому ймовірність того, що саккади досліджують периферію зображення, де, як очікується, буде знайдена деяка інформація, дуже мала. Це вказує на те, що модель захопила значну концентрацію інформації, що відноситься до класу, в середині зображень для цього конкретного набору для навчання.

У нижній частині малюнка 2.5 представлений огляд поведінки функції розпізнавання Href. Вихідна політика прогнозування порівнюється з (і) політикою

попиксельного підвищення різкості, яка вибирає карту з поточним припущенням zt, і (ii) рівномірним випадковим пошуком (випадковим чином вибирає наступну точку огляду). Показники класифікації, показані зліва, монотонно збільшуються зі зменшенням порога розпізнавання. Хоча швидкість розпізнавання становить 92%, що є верхньою межею (що відповідає вичерпного декодування з використанням 266 слабких класифікаторів - приблизно еквівалентно лінійному класифікатору), майже оптимальна швидкість розпізнавання виходить як для прогнозування, так і на основі попиксельного аналізу. -піксельная політика сегментації Href наближається до 1-5, в той час як випадкова політика явно недостатньо оптимальна. Додатковим ефектом є монотонне збільшення кількості мішків зі зменшенням Href, що показано на центральному та крайньому правому малюнку. Кількість мішків видатне за складністю розпізнавання. Розподіл кількості мішків у всіх випадках дуже суперечливо (центральний малюнок), невелика кількість мішків в більшості випадків відображає лише невелику частину, а багато мішки рідко відображають «недоступне» розпізнавання. Для політики прогнозування і політики попиксельного розпізнавання менше п'яти пакетів досить для досягнення порогу розпізнавання більш ніж в 50% випадків (в порівнянні з приблизно 15 в разі рандомізованого дослідження) для Href = 10-5.

Таким чином, сильним аспектом моделі є її здатність ефективно розпізнавати з дуже невеликою кількістю коефіцієнтів Хаара (і, отже, з дуже невеликою кількістю пікселів) в більшості випадків з невеликими витратами на обчислення, або з повною політикою прогнозування, або з попередньо обробленими банківськими картами і мішками. Кількість мішків відображає тривалість етапної обробки. Наприклад, середня кількість мішків від 10 до 15, коли Href = 1e-4, відповідає середньому стиску 85-90% даних, фактично оброблених для розпізнавання сцени. Число може бути вище, якщо поріг більш оптимістичний, і менше, якщо він більш консервативний.

Висновок за розділом

У цьому розділі ми розглянули, як нейронну мережу можна використовувати для вирішення завдань розпізнавання рукописного введення. Також було продемонстровано базовий алгоритм нейронної мережі з функцією динамічного прогнозування. Потім практичний підхід до застосування цього типу мережі був продемонстрований на прикладі розпізнавання рукописних чисел.

РОЗДІЛ 3
ЗАСОБИ РОЗРОБКИ СИСТЕМИ РОЗПАЗНАВАННЯ ТЕКСТУ

Подумайте про інструменти, методи та інших рішеннях, які оптимально відповідають вимогам, описаним раніше. У зв'язку з тим, що проект обмежений коротким терміном реалізації, було вирішено розглянути і проаналізувати найпопулярніші інструменти в своїй категорії.

Через цих функцій переважно зосередитися на інструментах обробки запитів або виконання програмного коду, які надають асинхронну операційну модель, засновану на подіях, або надають такі механізми. Це пов'язано з тим, що асинхронна модель умовно скорочує час відгуку програми.

3.1. Програмне середовище Tesseract

Tesseract (від англійського - «тессеракт») - безкоштовна комп'ютерна програма для розпізнавання тексту, розроблена Hewlett-Packard з середини 1980-х до середини 1990-х років і потім «пролежала на полиці» десять років. У серпні 2006 року Google купив його і випустив під ліцензією Apache 2.0, щоб продовжити розробку. На даний момент програма вже працює з UTF-8, мовна підтримка (в тому числі російський з версії 3.0) здійснюється за допомогою додаткових модулів.

Ядро програми Tesseract було розроблено в лабораторіях Hewlett Packard Bristol і Hewlett Packard Co., Greeley Colorado, 1985-1994 pp. У 1996 році були внесені істотні зміни і підготовлений порт для Windows. Потім, з 1998 року, частковий перехід з C на C ++. Велика частина коду спочатку була написана на C, але були внесені поліпшення в сумісність з компіляторами C ++.

Tesseract 3.0 в даний час побудований на Linux з GCC 2.95 і новіше і Windows з Visual C ++ 2008 Express і новіше (підтримка Visual C ++ 6 була видалена в версії 3.0).

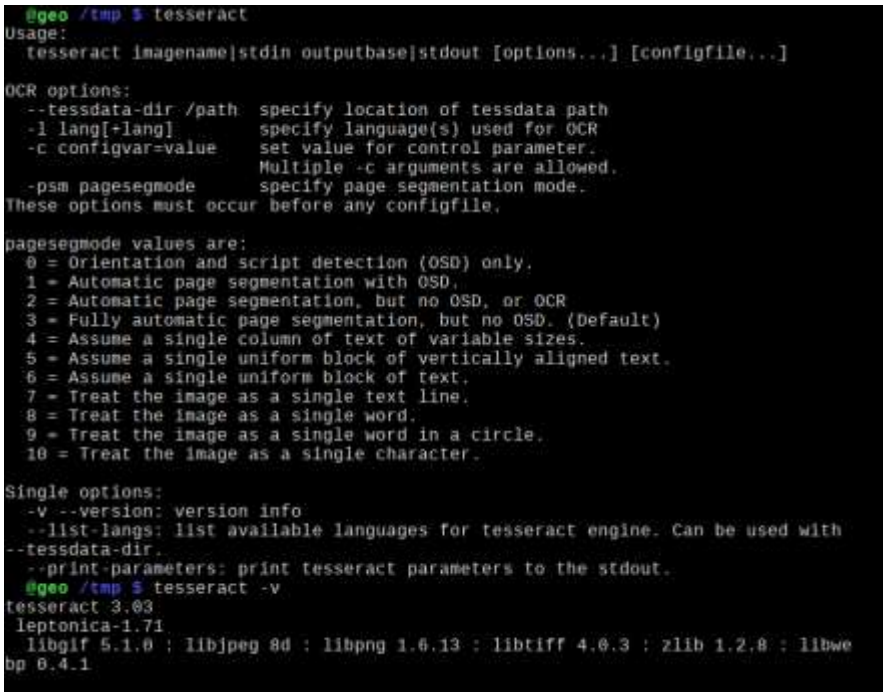

Рисунок 3.1 – Головне вікно програми

На даний момент останньою версією є Tesseract 4.0 на основі LSTM.

Як і всі інші ліцензії на безкоштовне програмне забезпечення, ліцензія Apache дає користувачеві право використовувати програмне забезпечення для будь-яких цілей, а також вільно змінювати і поширювати змінені копії в доповнення до назви.

Ця ліцензія не наказує незмінність ліцензії на поширення програмного забезпечення і навіть не вимагає збереження її безкоштовного і відкритого статусу. Єдина умова, що накладається ліцензією Apache, - інформувати одержувача про використання вихідного коду. На відміну від ліцензій з авторським лівому, одержувач зміненої версії не обов'язково має всі права, в першу чергу, з ліцензією Apache.

При поширенні програмного забезпечення ви повинні помістити в кореневий каталог наступні файли:

ЛІЦЕНЗІЯ - файл, що містить копію тексту ліцензії Apache;

ПРИМІТКА - це текстовий файл, в якому перераховані всі бібліотеки, ліцензовані за ліцензією Apache, а також імена їх творців.

Кожен файл, уповноважений робити це, повинен містити всю вихідну інформацію про авторські права або патенти, і кожен змінений файл повинен супроводжуватися інформацією про внесені зміни.

3.2. Формат файлу збереження

Portable Document Format (PDF) - це відкритий для платформи формат електронного документа, спочатку розроблений Adobe Systems з рядом функцій мови PostScript. Він в першу чергу призначений для презентації поліграфічної продукції в електронному вигляді. Існує безліч програм для перегляду, наприклад офіційна безкоштовна програма Adobe Reader. Велика кількість сучасного професійного поліграфічного обладнання має апаратну підтримку формату PDF, тому ви можете друкувати документи в цьому форматі без використання будь-якого програмного забезпечення. Традиційним способом створення PDF-документів є віртуальний принтер, тобто документ готується в своїй спеціальній програмі - графічній програмі або текстовому редакторі, САПР і т. Д., А потім експортується в формат PDF для електронного поширення, передачі в друкарню. і ін. П.

PDF є відкритим стандартом ISO 32000 з 1 липня 2008 року.

Формат PDF дозволяє вставляти необхідні шрифти (простий текст), векторні і растрові зображення, форми і мультимедійні вставки. Підтримує RGB, CMYK, Grayscale, Lab, Duotone, Bitmap, кілька типів стиснення растрових зображень. Має власні технічні формати для друку: PDF / X-1a, PDF / X-3. Включає механізм електронного підпису для захисту і аутентифікації документів. У цьому форматі поширюється велика кількість супутньої документації.

Найчастіше файл PDF являє собою комбінацію тексту з растровою і векторною графікою, рідше текст з фігурами, скрипти JavaScript, 3D-графіку і інші типи елементів.

Обсяги інформації для двох, які виглядають однаково на екрані документів PDF, можуть значно відрізнятися в залежності від:

- вставляти або пов'язувати шрифти і мультимедіа;

- дозвіл растрових зображень;

- використовує вбудований механізм стиснення для всього документа;

- використовувати алгоритми стиснення растрових зображень.

Щоб створити документ мінімального розміру, необхідно використовувати векторну графіку і «безпечні» шрифти. Всього таких шрифтів 14:

- Times (v3) (звичайний, курсив, напівжирний і напівжирний курсив)

Кур'єр (звичайний, курсив, напівжирний і напівжирний курсив)

- Helvetica (v3) (звичайний, курсив, напівжирний і напівжирний курсив)

- Символ [sv]

- Zapf Dingbats [ru]

Ці шрифти можна використовувати без вбудовування в документ, оскільки всі програми повинні їх правильно відображати. Всі інші шрифти, вбудовані в документ і відсутні в системі, будуть замінені одним з існуючих, що може привести до неправильних результатам пошуку сторінок, кількості символів в рядку і інших помилок відображення, пов'язаних зі шрифтами.

Існує технологія MRC (Mixed Raster Content), яка наближає функції PDF до можливостей DjVu зі зберігання відсканованих зображень з текстом.

3.3. Технологія для розробки інтерфейсу – Java

Java - дуже типовий об'єктно-орієнтована мова програмування загального призначення, розроблений Sun Microsystems (в подальшому придбаний Oracle). Розробка виконується спільнотою, організованим в рамках процесу спільноти Java; Speech і базова реалізація його технології поширюються за ліцензією GPL. Права на товарний знак належать Oracle Corporation.

Програми Java зазвичай переводяться в спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі, якщо існує реалізація

віртуальної машини Java. Офіційна дата випуску - 23 травня 1995 року. Станом на 2019 рік Java є одним з найпопулярніших мов програмування.

Мова спочатку називалася Oak (Дуб) і була розроблена Джеймсом Гослінгом для програмування пристроїв побутової електроніки. Оскільки номер з такою назвою вже існував, Oak був перейменований в Java. Названий на честь кавового бренду Java, який в свою чергу отримав назву однойменного острова (Ява), тому на офіційній емблемі мови зображена чашка гарячої кави. Існує ще одна версія походження назви мови, пов'язана з натяком на кавоварку як приклад побутового приладу, для програмування якого вперше була створена мова. Згідно етимології російськомовної літератури з кінця двадцятого до перших років XXI століття, назва мови часто перекладається як ява і не транскрибувати.

В результаті цього проекту світ побачив абсолютно новий пристрій - кишеньковий комп'ютер Star7, який випередив свій час більш ніж на десять років, але через свою високу вартість в 50 доларів він не зміг зробити революцію в світі технологій і був забули.

Пристрій Star7 було не так популярно, як мова програмування Java і його середовище. Наступним кроком в житті мови стало розвиток інтерактивного телебачення. У 1994 році стало ясно, що інтерактивне телебачення було помилкою.

З середини 1990-х років мова широко використовувалася для написання клієнтських додатків і серверного програмного забезпечення. У той же час певного поширення набула технологія Java-аплетів - графічних додатків Java, вбудованих в веб-сторінки; з появою 2000 року динамічних функцій веб-сторінок технологія використовувалася рідко.

Spring Framework використовується в веб-розробці; Інструмент Javadoc використовується для документації.

Програми Java переводяться в байт-код Java, який запускається віртуальною машиною Java (JVM), програмою, яка обробляє байт-код і відправляє інструкції обладнанню як інтерпретатора.

Перевага цього методу запуску програм полягає в повній незалежності від коду змін операційної системи і устаткування, що дозволяє запускати Java-додатки на будь-якому пристрої, де є відповідна віртуальна машина. Ще одна

важлива особливість технології Java - гнучка система безпеки, в якій програма повністю запускається віртуальною машиною. Будь-яка дія, що перевищує встановлені дозволу програми (наприклад, спроба несанкціонованого доступу до даних або підключення до іншого комп'ютера), викликає негайне переривання.

До недоліків концепції віртуальної машини часто можна віднести зниження продуктивності. Ряд поліпшень трохи збільшили швидкість запуску Java-додатків:

-застосування технології перекладу коду змін в машинний код безпосередньо під час роботи програми (технологія JIT) з можливістю збереження версій класу в машинному коді,

Широке застосування переноситься орієнтованого коду (нативний код) в стандартних бібліотеках,

обладнання, що забезпечує швидку обробку байтового коду (наприклад, технологія Jazelle, яка підтримується деякими процесорами ARM).

Згідно Shotout.alioth.debian.org, час виконання семи різних завдань на Java в середньому в півтора-два рази довше, ніж для C / C ++, в деяких випадках Java працює швидше, а в деяких випадках В 7 разів повільніше. З іншого боку, для більшості з них споживання пам'яті Java-машиною було в 10-30 разів більше, ніж у програми C / C ++. Також варто відзначити дослідження, проведене Google, згідно з яким в тестових прикладах на Java значно нижче продуктивність і вище споживання пам'яті в порівнянні з аналогічними програмами на C ++.

Ідеї, що лежать в основі концепції і різних реалізацій середовища віртуальної машини Java, надихнули багатьох ентузіастів розширити список мов, які можна використовувати для створення додатків, що працюють на віртуальній машині. Ці ідеї також виражені в специфікації CLI, яка є основою платформи .NET від Microsoft.

Мова Java активно використовується для створення мобільних додатків для операційної системи Android. В цьому випадку програми компілюються в нестандартний код обміну для використання віртуальної машини Dalvik (починаючи з Android 5.0 Lollipop віртуальна машина буде замінена на ART). Для такої компіляції використовується додатковий інструмент, а саме Android SDK (Software Development Kit), розроблений Google.

Розробка додатків може виконуватися в Android Studio, NetBeans, Eclipse за допомогою програми-надбудови Android Development Tools (ADT) або в IntelliJ IDEA. Версія JDK повинна бути 5.0 або вище.

8 грудня 2014 р Android Studio визнана Google офіційної середовищем розробки для ОС Android.

Висновок за розділом

У цьому розділі були представлені застереження щодо вибору інструменту для створення додатка для розпізнавання символів. Описуються кращі аспекти використання обраних рішень, описується весь програмний продукт і способи його використання.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДОДАТКУ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ OCR

4.1. Специфікація програмного забезпечення

Давайте почнемо розробку програмного забезпечення з опису призначених для користувача ролей, які будуть створені під час реалізації, і сценаріїв використання системи для цих ролей. У додатку передбачено такі ролі користувачів, як адміністратор, модератор, користувач, гість і користувачі, яким заблокований доступ до додатка.

Рис.4.1. Варіанти використання додатку для користувача

Користувач (рисунок 4.1) може при установці програми зареєструватися та увійти в систему або продовжити роботу в якості гостя з обмеженою функціональністю.

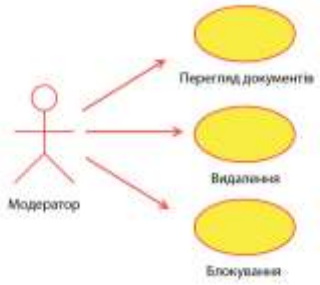

Рис.4.2. Варіанти використання додатку для модератора

Модератор (рисунок 4.2) має можливість переглядати документи користувачів, видаляти документи і переміщати користувачів в «заблоковану» групу в разі зловживань.

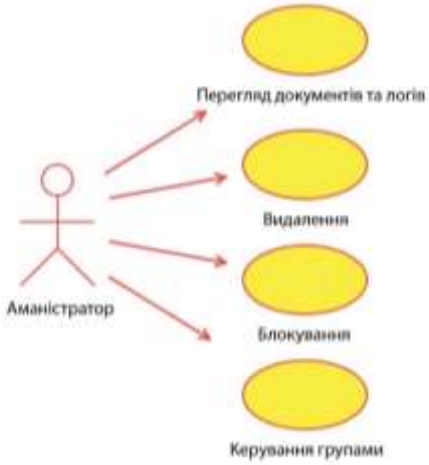

Рис.4.3. Варіанти використання додатку для адміністратора

Адміністратор (рисунок 4.3) має можливість переглядати документи користувачів, переглядати журнали додатків, а також керувати групами користувачів.

Рис.4.4. Варіанти використання додатку для заблокованого користувача

Користувач, якого заблокований доступ до додатка, може сканувати документи тільки локально без використання хмарних технологій.

4.2. Алгоритм роботи додатку

Коротко опишемо процес роботи з системою для користувачів, що використовують такий алгоритм.

Користувач за допомогою програми і камери, встановленої в пристрої, сканує документ, який необхідно розпізнати. Додаток автоматично використовує бібліотеку Tesseract-ocr, локальну обробку документа, визначає межі тексту, зображень та аналізує інші параметри відсканованого документа.

Додаток використовує внутрішній алгоритм Tesseract ocr для приблизного визначення точності локального розпізнавання. Якщо точність розпізнавання висока, робота в місцевій бібліотеці буде продовжена. Якщо точність розпізнавання не відповідає бажаному рівню, документ розпізнавання відправляється в хмарний сервіс. Система працює з API.

Бібліотека Tesseract ocr містить пакет OCR libtesseract і програму командного рядка tesseract. Tesseract 5 надає новий механізм OCR (LSTM) на основі нейронних мереж з акцентом на розпізнавання рядків, але як і раніше підтримує більш старий механізм розпізнавання Tesseract 3, який працює шляхом розпізнавання шаблонів символів.

Tesseract підтримує Unicode (UTF-8) і миттєво розпізнає понад 100 мов.

Tesseract підтримує різні формати виводу: простий текст, hOCR (HTML), PDF, PDF, нетекстовий формат, TSV. Щоб отримати найкращі результати OCR, у багатьох випадках вам необхідно поліпшити якість вхідного зображення за допомогою додаткових хмарних сервісів.

4.3. Розробка та підключення до бази даних

JDBC розроблений як альтернатива API-інтерфейсу ODBC (Open Database Connectivity) на основі C і пропонує інтерфейс на рівні програмного забезпечення, який керує механізмами додатків Java, які взаємодіють з базою даних або СУБД. Інтерфейс JDBC має два рівні:

JDBC API (рис. 4.5) підтримує зв'язок між додатком Java і менеджером JDBC.

Драйвер JDBC підтримує зв'язок між менеджером JDBC і драйвером бази даних.

JDBC - це загальний API, з яким взаємодіє ваш програмний код. Нижче наведено JDBC-сумісний драйвер для використовуваної бази даних.

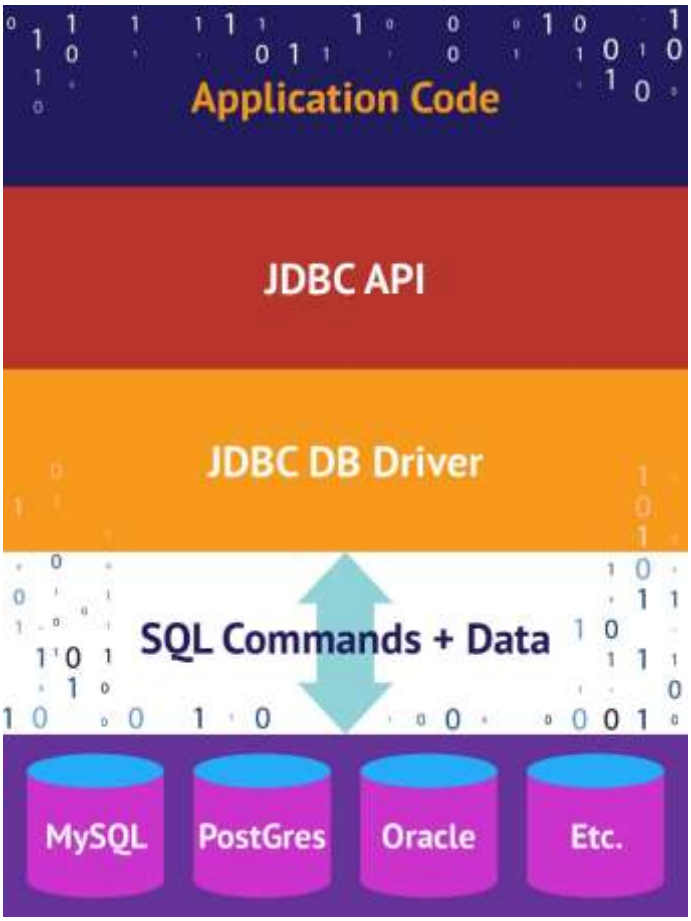

Рис. 4.5. Архітектурний огляд JDBC на рівні стійкості Java

Кожен з цих операцій імпорту надає доступ до класу, який спрощує стандартні підключення до бази даних Java:

З'єднання є з'єднання з базою даних.

DriverManager отримує підключення до бази даних. (Інший варіант - DataSource, який використовується для збору з'єднань.)

SQLException обробляє помилки SQL між додатком Java і базою даних.

Результати даних ResultSet і Statement моделі і оператори SQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;
```

Рис. 4.6. Імпорт пакетів для підключення до БД

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;

class WhatIsJdbc{
  public static void main(String[] args) {
    Connection conn = null;
    try {
      String url = "jdbc:sqlite:path-to-db/chinook/chinook.db";
      conn = DriverManager.getConnection(url);

      System.out.println("Got it!");

    } catch (SQLException e) {
        throw new Error("Problem", e);
    } finally {
      try {
        if (conn != null) {
            conn.close();
        }
      } catch (SQLException ex) {
          System.out.println(ex.getMessage());
      }
    }
  }
}
```

Рис. 4.7. Розробка класу підключення до бази даних

4.4. Створення *API*

Tesseract можна використовувати безпосередньо через командний рядок або за допомогою API (рис. 4.8) для вилучення друкованого тексту із зображень. Він підтримує широкий спектр мов.

```
#if defined(_WIN32)
#include <fcntl.h>
#include <io.h>
#else
#include <dirent.h>            // for closedir, opendir, readdir, DIR, dirent
#include <libgen.h>
#include <sys/types.h>
#include <sys/stat.h>          // for stat, S_IFDIR
#include <unistd.h>
#endif  // _WIN32

#include <cmath>               // for round, M_PI
#include <cstdint>             // for int32_t
#include <cstring>             // for strcmp, strcpy
#include <fstream>             // for size_t
#include <iostream>            // for std::cin
#include <locale>              // for std::locale::classic
#include <memory>              // for std::unique_ptr
#include <set>                 // for std::pair
#include <sstream>             // for std::stringstream
#include <vector>              // for std::vector
#ifdef HAVE_LIBCURL
#include <curl/curl.h>
#endif
#include "allheaders.h"        // for pixDestroy, boxCreate, boxaAddBox, box...
#ifndef DISABLED_LEGACY_ENGINE
#include "blobclass.h"         // for ExtractFontName
#endif
#include "boxword.h"           // for BoxWord
#include "coutln.h"            // for C_OUTLINE_IT, C_OUTLINE_LIST
#include "dawg_cache.h"        // for DawgCache
#include "dict.h"              // for Dict
#include "edgblob.h"           // for extract_edges
#include "elst.h"              // for ELIST_ITERATOR, ELISTIZE, ELISTIZEH
#include "environ.h"           // for l_uint8
#ifndef DISABLED_LEGACY_ENGINE
#include "equationdetect.h"    // for EquationDetect
#endif
#include "errcode.h"           // for ASSERT_HOST
#include <tesseract/helpers.h>          // for IntCastRounded, chomp_string
#include "imageio.h"           // for IFF_TIFF_G4, IFF_TIFF, IFF_TIFF_G3, ...
#ifndef DISABLED_LEGACY_ENGINE
#include "intfx.h"             // for INT_FX_RESULT_STRUCT
```

Рис.4.8. Робота з API у бібліотеці tessercat

```
int TessBaseAPI::Recognize(ETEXT_DESC* monitor) {
  if (tesseract_ == nullptr)
    return -1;
  if (FindLines() != 0)
    return -1;
  delete page_res_;
  if (block_list_->empty()) {
    page_res_ = new PAGE_RES(false, block_list_,
                             &tesseract_->prev_word_best_choice_);
    return 0; // Empty page.
  }
```

Рис.4.9. API для отримання результатів розпізнавання

OCRmyPDF - додає шар OCR тексту в відсканованої PDF-файли і зображення, щоб ви могли їх шукати. Обробляє сторінки паралельно на багатоядерних процесорах. По можливості зберігає точне дозвіл вихідних вбудованих зображень без стиснення JPEG. Містить кілька варіантів попередньої обробки зображень, детальну документацію і підтримку багатьох екзотичних PDF-файлів.

pdf2pdfocr - це інструмент для розпізнавання PDF (або підтримуваних зображень) і додавання шару тексту в вихідний файл, що робить його доступним для пошуку PDF. Це скрипт Python, який використовується tesseract і іншими інструментами з відкритим вихідним кодом. Підтримуються Linux, macOS і Windows.

pdf2searchablepdf - це інструмент, який дозволяє конвертувати будь-який PDF-файл, в якому неможливо виконати пошук, АБО будь каталог з зображеннями цілком, в PDF-файл, в якому можна виконувати пошук.

```
char* TessPDFRenderer::GetPDFTextObjects(TessBaseAPI* api,
                                         double width, double height) {
  double ppi = api->GetSourceYResolution();

  // These initial conditions are all arbitrary and will be overwritten
  double old_x = 0.0, old_y = 0.0;
  int old_fontsize = 0;
  tesseract::WritingDirection old_writing_direction =
      WRITING_DIRECTION_LEFT_TO_RIGHT;
  bool new_block = true;
  int fontsize = 0;
  double a = 1;
  double b = 0;
  double c = 0;
  double d = 1;

  std::stringstream pdf_str;
  // Use "C" locale (needed for double values prec()).
  pdf_str.imbue(std::locale::classic());
  // Use 8 digits for double values.
  pdf_str.precision(8);

  // TODO(jbreiden) This marries the text and image together.
  // Slightly cleaner from an abstraction standpoint if this were to
  // live inside a separate text object.
  pdf_str << "q " << prec(width) << " 0 0 " << prec(height) << " 0 0 cm";
  if (!textonly_) {
    pdf_str << " /Im1 Do";
  }
  pdf_str << " Q\n";

  int line_x1 = 0;
  int line_y1 = 0;
  int line_x2 = 0;
  int line_y2 = 0;

  ResultIterator *res_it = api->GetIterator();
  while (!res_it->Empty(RIL_BLOCK)) {
    if (res_it->IsAtBeginningOf(RIL_BLOCK)) {
      pdf_str << "BT\n3 Tr";     // Begin text object, use invisible ink
      old_fontsize = 0;          // Every block will declare its fontsize
      new_block = true;          // Every block will declare its affine matrix
    }
```

Рис.4.10. Приклад коду роботи модулю відображення PDF

4.5. Створення інтерфейсу додатку

Інтерфейс (UI) Android-додатки побудований у вигляді ієрархії макетів і віджетів. Макети - це об'єкти ViewGroup (рис. 4.11), контейнери, які керують розміщенням своїх дочірніх уявлень на екрані. Віджети - це об'єкти перегляду, компоненти інтерфейсу, такі як кнопки і текстові поля.

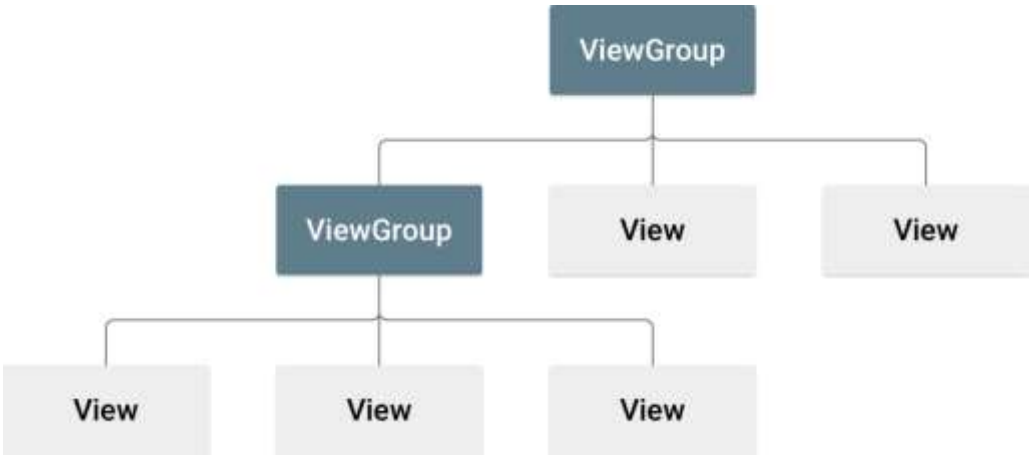

Рис.4.11 Ілюстрація того, як об’єкти ViewGroup утворюють гілки в макеті та містять об’єкти View.

Android пропонує словник XML для класів ViewGroup і View, тому більша частина вашого інтерфейсу визначена в файлах XML. Замість того, щоб навчитися писати XML, це керівництво покаже вам, як створити макет за допомогою редактора макетів Android Studio. Редактор макета записує XML для вас, коли ви перетягуєте уявлення для створення макета.

У вікні проекту відкрийте додаток> res> layout> activity_main.xml.

Сховайте вікно проекту, щоб звільнити місце для редактора макета. Для цього виберіть «Перегляд»> «Інструменти Windows»> «Проекти» або просто натисніть «Проекти» в лівій частині екрана Android Studio.

Якщо ваш редактор відображає джерело XML, клацніть вкладку «Дизайн» в нижній частині вікна.

Натисніть «Вибрати область дизайну» і виберіть «Blueprint».

Клацніть «Перегляд» на панелі інструментів редактора макета і переконайтеся, що ви обрали «Показати всі обмеження».

Переконайтеся, що автомобільний зарядний пристрій вимкнений. Підказки на панелі інструментів відображають «Активувати автоматичне з'єднання з батьками», коли автоматичне з'єднання вимкнено.

Клацніть Поля за замовчуванням на панелі інструментів і виберіть 16. При необхідності ви можете налаштувати поля для кожного уявлення пізніше.

Клацніть «Попередній перегляд пристрою» на панелі інструментів і виберіть 5,5, 1440 × 2560, 560 точок на дюйм (Pixel XL).

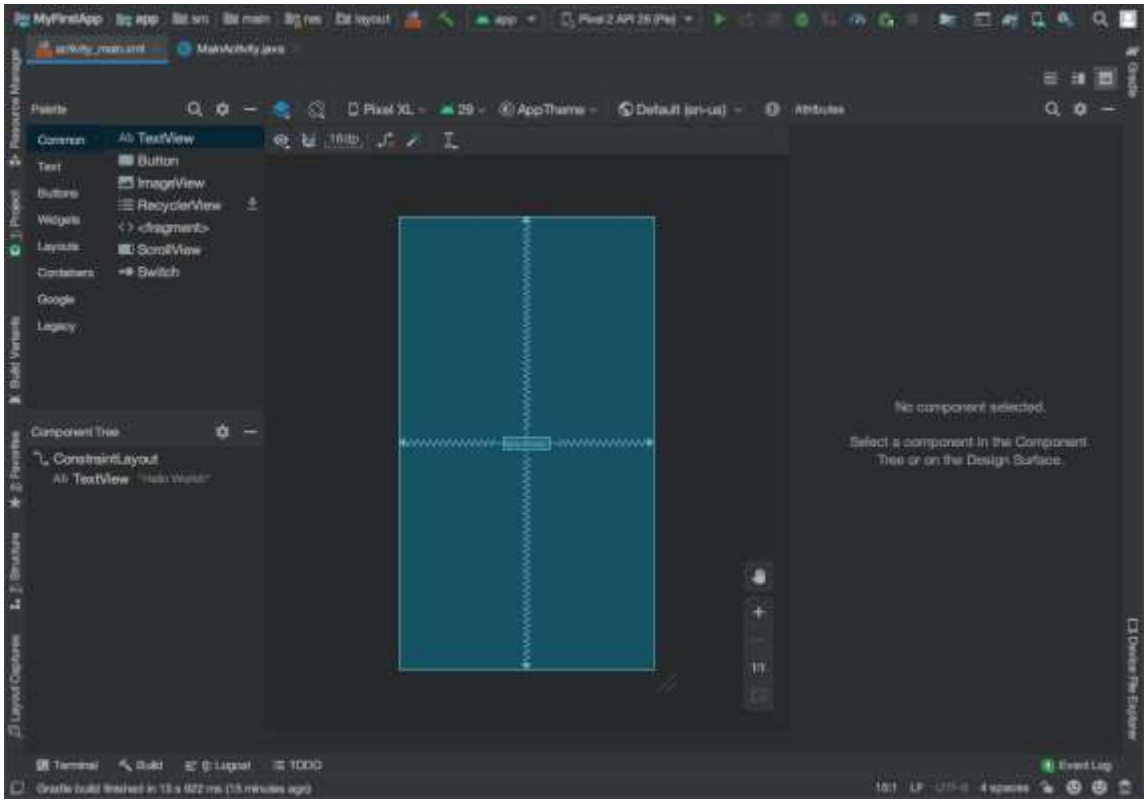

Рис.4.12. Редактор макета, що відображає Activity_main.xml

На панелі «Дерево компонентів» в лівому нижньому куті екрана ієрархія уявлення макета. В цьому випадку кореневих поданням є ConstraintLayout, який містить тільки один об'єкт TextView.

ConstraintLayout - це макет, який визначає положення кожного уявлення на основі обмежень родинних і батьківських уявлень макета. Таким чином, ви можете створювати як прості, так і складні макети з плоскою ієрархією презентацій. Цей тип макета дозволяє уникнути вкладених макетів. Інкапсульований макет, який є

прототипом макета, показаного на малюнку 2, може збільшити час, необхідний для створення призначеного для користувача інтерфейсу.

Для початку потрібно видалити те, що вже є в макеті. Клацніть TextView у вікні «Дерево компонентів», а потім натисніть клавішу «Видалити».

Клацніть Текст на панелі палітри, щоб переглянути доступні об'єкти управління текстом.

Перетягніть простий текст в редактор дизайну і помістіть його вгорі макета. Це віджет EditText, який приймає простий текстовий введення.

Клацніть вид в редакторі дизайну. Тепер ви можете бачити квадратні ручки для зміни розміру виду в кожному кутку і круглі примусові якоря з кожного боку. Для кращого контролю ви можете збільшити масштаб редактора. Для цього використовуйте кнопки масштабування на панелі інструментів редактора макета.

Натисніть і утримуйте якір вгорі, потягніть його вгору, поки він не зафіксується в верхній частині макета, а потім відпустіть. Це обмеження: воно обмежує вид в межах встановленого поля за замовчуванням. В цьому випадку встановіть значення 16 dp від верхнього краю макету.

Використовуйте той же процес, щоб створити обмеження з лівого боку виду на ліву сторону макета.

Натисніть кнопку на панелі палітри.

Перетягніть віджет кнопки в редакторі дизайну і опустіть його вправо.

Створіть обмеження з лівого боку кнопки праворуч від текстового поля.

Створіть обмеження між базовими лініями тексту, щоб обмежити потік в горизонтальному напрямку. Для цього клацніть правою кнопкою миші і виберіть «Показати базову лінію» «Показати дію базової лінії» в редакторі макета. Основні прив'язки відображаються всередині кнопки. Клацніть і утримуйте цю прив'язку і перетягніть її на підставу прив'язки, показане в сусідньому текстовому полі.

Для попереднього перегляду інтерфейсу натисніть «Вибрати область дизайну» на панелі інструментів і виберіть «Дизайн». Зверніть увагу, що для введення тексту і мітки кнопки задані настройки за замовчуванням.

Виконайте наступні дії, щоб змінити рядки інтерфейсу:

Відкрийте вікно проекту, а потім відкрийте програму> res> values> strings.xml.

Це файл строкових ресурсів, в якому ви можете вказати всі рядки інтерфейсу. Це дозволяє вам керувати всіма рядками призначеного для користувача інтерфейсу в одному місці, що спрощує їх пошук, оновлення і розміщення.

Натисніть «Відкрити редактор» вгорі вікна. Відкриється редактор перекладу, який надає простий інтерфейс для додавання і редагування стандартних рядків. Це також допомагає організувати все перекладені рядки.

Клацніть Додати ключ, щоб створити новий рядок як «текстовий текст» для текстового поля.

У діалоговому вікні «Додати ключ» виконайте наступні дії (рис. 4.13):

Введіть «edit_message» в поле «Ключ».

Введіть «введіть повідомлення» в поле «Значення за замовчуванням».

Клацніть ОК.

Додайте ключ з ім'ям «button_send» зі значенням «Відправити».

Тепер ви можете встановити ці рядки для кожного виду. Щоб повернутися до файлу макета, клацніть Activity_main.xml на панелі вкладок. Потім додайте такі рядки:

Клацніть у текстовому полі в макеті. Якщо вікно атрибутів ще не відображається праворуч, клацніть «Атрибути» на правій бічній панелі.

Знайдіть властивість тексту, яке в даний час встановлено на Ім'я, і видаліть значення.

Знайдіть властивість підказки, натисніть «Вибрати ресурс», розташоване праворуч від текстового поля. З'явиться діалогове вікно, двічі клацніть edit_message зі списку.

Натисніть кнопку в макеті і знайдіть її текстовий властивість, для якого тепер встановлено значення «Кнопка». Клацніть Вибрати ресурс і виберіть button_send.

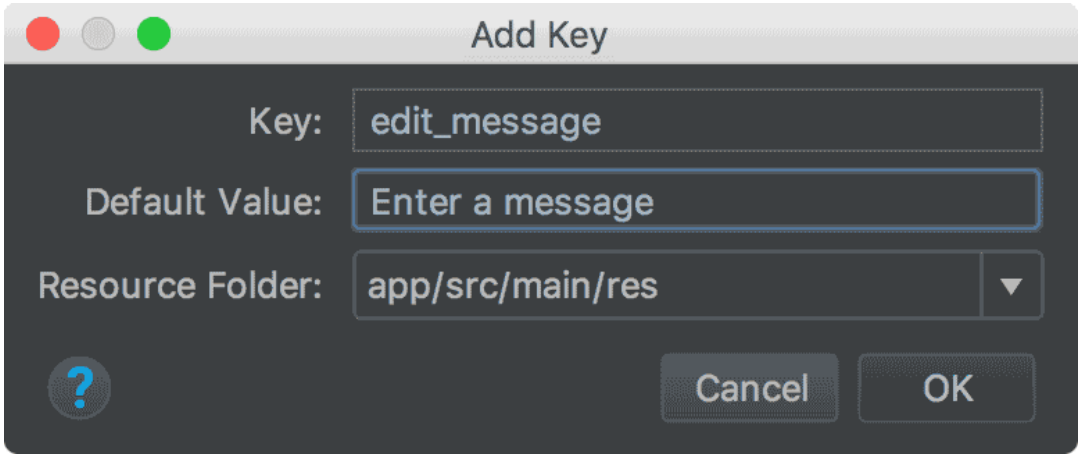

Рис.4.13. Діалогове вікно для додавання нового рядка

4.6. Взаємодія користувача з додатком

Для запуску додатку необхідно його завантажити та встановити. Наприклад, як показано на рис. 4.13.

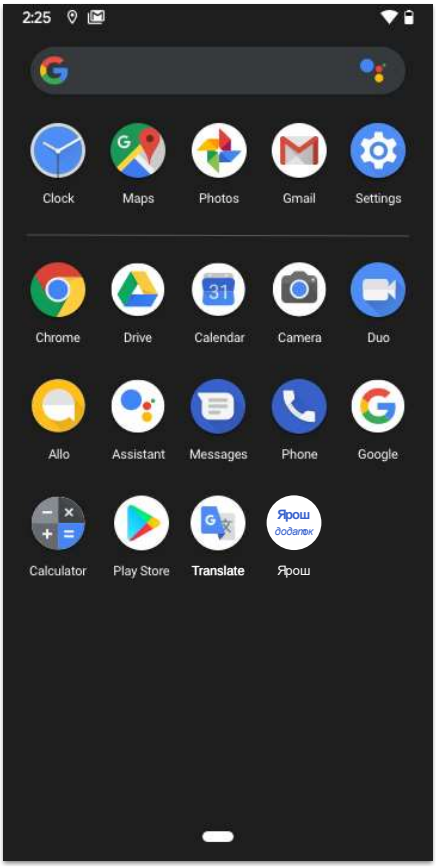

Рис.4.14. Виклик додатку в смартфоні

Після виклику додатку відкриється головна сторінка, яка представлена на рис. 4.15.

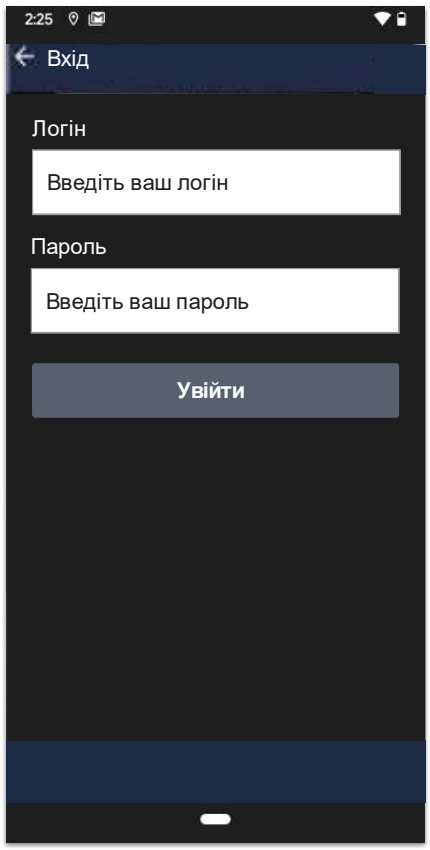

Рис.4.15. Екран авторизації в додатку

Також можливо зареєструватися в додатку не вихоячи з нього. Приклад реєстрації представлений на рис.4.16.

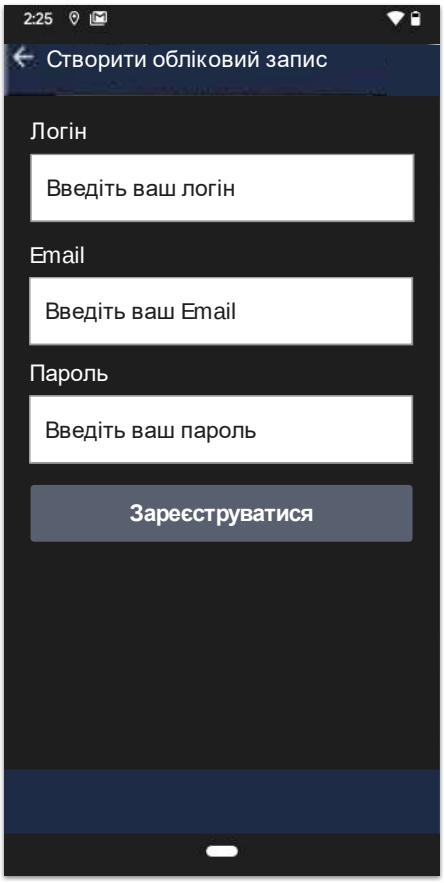

Рис.4.16. Екран створення акаунту в додатку

Після авторизації можна переглянути документи які були вісканові раніше, за допомогою збереження на “хмарі” (рис. 4.17).

Рис.4.17. Галерея раніше розпізнаних документів

Додаток пітримую збереження як доданих до нього зображень, так і збережених текстів які були відскановані з зображення з можливістю редагування, копіювання та виалення непотрібних часток тексту (рис. 4.18).

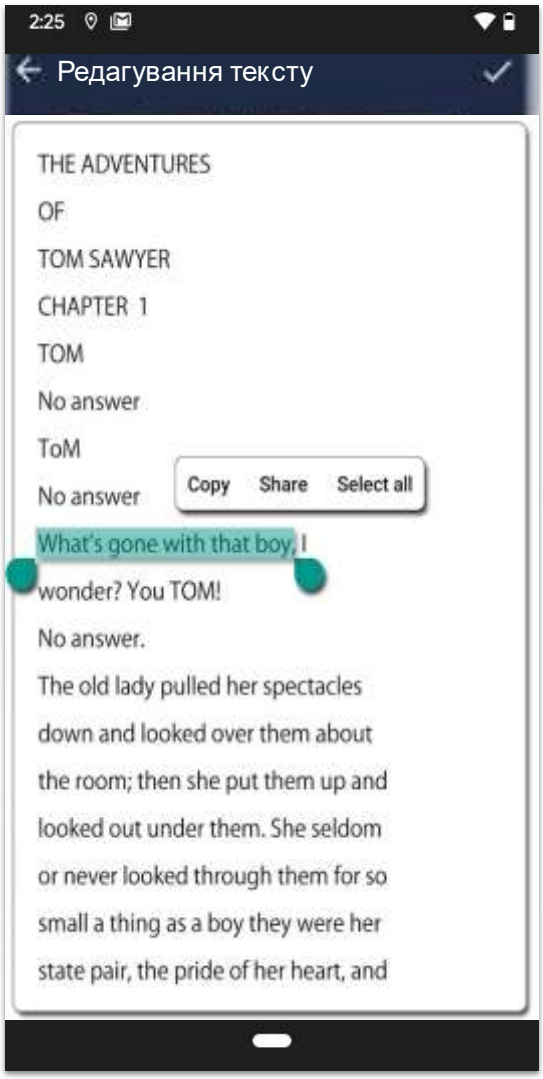

Рис.4.18. Перегляд тексту розпізнаного документу з можливістю редагування

4.7. Проведення експериментів та аналіз отриманих результатів

Для експериментів було обрано розпізнавання тексту в двох категоріях для двох типів тексту: для розпізнавання 1-2 пропозицій і для розпізнавання всієї сторінки тексту в друкованих та рукописних текстах. Всі цифрові зображення були взяті з друкованих сторінок, а не з цифрових пристроїв, щоб поліпшити якість розпізнавання. Всі текстові копії мали різні шрифти, що дозволяло оцінити універсальність програми.

4.7.1. Результат розпізнавання малих частин рукописного тексту

Для цього експерименту були взяті 20 фрагментів тексту, кожне з яких складається з одного або двох пропозицій, кількість загальних символів і кількість неправильних символів були записані і зібрані в таблиці (Таблиця 4.1).

Таблиця 4.1

№ експерименту	К-ть символів	К-ть помилок	% розпізнавання
1	108	1	99,3
2	95	5	93,7
3	115	5	96,2
4	120	3	97,5
5	98	4	94,9
6	80	0	100
7	105	4	95,3
8	80	2	97,5
9	101	13	88,10
10	118	4	95,8
11	101	7	93,2
12	94	2	97,9
13	99	5	93,9
14	101	10	90,7
15	121	6	95,1
16	102	8	92,3
17	92	2	97,9
18	105	5	97,1
19	100	7	93,2
20	128	2	98,4

Після отриманих результатів можна зробити висновок, що система досить добре розпізнає невеликі фрагменти тексту із середнім результатом трохи краще 95% (95,42). І завжди показує результат краще 85%. Тобто системі досить просто призначити невеликі фрагменти тексту для розпізнавання, а потім дати користувачеві, щоб він трохи змінив результати, щоб отримати повністю правильні дані.

4.7.2. Результат розпізнавання великих частин рукописного тексту

Для цього експерименту було взято 20 фрагментів тексту, півсторінки або більше, кількість загальних символів і кількість неправильних символів були записані і зібрані в таблиці (Таблиця 4.2).

Таблиця 4.2

№ експерименту	К-ть символів	К-ть помилок	% розпізнавання
1	893	12	98,6
2	620	18	97,1
3	1025	40	96,1
4	885	81	90,8
5	902	52	94,2
6	855	38	95,5
7	1017	119	88,2
8	700	97	86,1
9	892	53	94,1
10	1047	165	84,2
11	1208	188	84,4
12	680	51	92,5
13	787	160	79,6
14	826	165	80,1
15	895	66	92,6

16	905	315	65,1
17	967	182	81,2
18	803	61	92,4
19	711	89	87,4
20	1020	195	80,9

За отриманими результатами можна зробити висновок, що система розпізнає великі частини тексту трохи гірше з середнім результатом трохи краще 88% (88,06). І досить часто цей результат виявляється краще 80%. Тобто системі можуть бути призначені великі шматки тексту, але користувач повинен обов'язково виправити від 10 до 20 відсотків помилок, щоб отримати правильний текст на виході.

4.7.3. Результат розпізнавання малих частин друкованого тексту

Для цього експерименту було взято 20 фрагментів тексту, одне або два пропозиції в кожному, кількість загальних символів і кількість неправильних символів були записані і зібрані в таблиці (Таблиця 4.3).

Таблиця 4.3

№ експерименту	К-ть символів	К-ть помилок	% розпізнавання
1	108	1	99,1
2	95	2	97,8
3	115	2	98,4
4	120	3	97,4
5	98	1	98,9
6	80	0	100
7	105	0	100
8	80	2	97,4
9	101	1	99,5
10	118	0	100

11	101	4	96,2
12	94	2	92,7
13	99	2	96,8
14	101	1	99,8
15	121	1	99,9
16	102	1	99,7
17	92	2	97,9
18	105	0	100
19	100	3	97,4
20	128	2	98,3

За отриманими результатами можна зробити висновок, що система досить добре розпізнає невеликі фрагменти тексту із середнім результатом трохи краще 98% (98,32). І завжди показує результат краще 95%. Тобто системі досить просто призначити невеликі фрагменти тексту для розпізнавання, а потім дати користувачеві, щоб він трохи змінив результати, щоб отримати повністю правильні дані.

Висновок за розділом

У цьому розділі представлені попередження при виборі інструменту для створення додатка для розпізнавання символів. Описано кращі аспекти використання обраних рішень. Описано весь програмний продукт і способи його використання. Також були представлені результати програми, які відбили, що система досить добре справляється з невеликими фрагментами рукописного тексту, друкованого тексту і трохи гірше обробляє великі.

Четвертий розділ присвячений дизайну і розробки програми. Дизайн додатка було розпочато з опису ролей користувачів і представлений у вигляді діаграм UML. Діаграми дій є опис варіантів використання. Певні варіанти використання, ролі і робочі моделі в системі дозволяють програмісту повністю уявити бачення і

реалізувати всі намічені вимоги. Також був розроблений і описаний алгоритм системи.

Описано і представлені функціональні можливості програмного забезпечення для роботи з ним користувача.

ВИСНОВКИ

У цій роботі був розроблений програмний продукт для компонентів робочого столу, який дозволяє користувачеві розпізнавати текст по символам з п'яти різних мов, використовуючи модель, створену за допомогою алгоритму динамічного прогнозування.

Отримана система показала непогані результати в розпізнаванні тексту. Для фотографій, що містять близько ста символів, середній результат розпізнавання становив 95 відсотків, а мінімальний показник завжди досягав значення вище 85 відсотків.

Система непогано справляється з дрібними частинами рукописного тексту, письмового тексту і ще гірше - з великими.

Подальшим дослідженням для цієї роботи може бути реалізація цього алгоритму для задач розпізнавання підпису.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Герберт Ф. OCR: історія, розпізнавання символів optisk / Герберт Ф. Шнц - Манчестер: Манчестерський центр, Вт., 1982 - 156 с.

2. Терміни OCR [Електронний ресурс] - 2018. - Режим доступу до ресурсів: https://en.wikipedia.org/wiki/Timeline_of_optical_character_recognition

3. HPE Haven OnDemand [Електронний ресурс] - 2015. - Режим доступу до ресурсу: http://www.havenondemand.com

4. OCR на Google Glass [Електронний ресурс] - 2014. - Режим доступу до ресурсу: https://community.havenondemand.com/t5/Blog/Tutorial-OCR-on-Google-Glass/ba-p/1164

5. Фомін, С.В. Математичні проблеми біології / С.В. Фомін, М.Б. Беркінбліт - М.: Наука, 1973 - 200 с.

6. Розенблатт Ф. Принципи нейродинаміки: персептрони і теорія мозкових механізмів / Френк Розенблат; пер. з англ. В.Я. Алтаєв. - М.: Світ, 1965 - 480 с.

Брюхоміцкій Ю.А. Моделі нейромереж для систем захисту інформації / Юрій Брюхоміцкій - Таганрог: ТРТУ, 2005. - 160 с.

8. Вассерман Ф. Нейрокомп'ютерні технології: теорія і практика / Френсіс Вассерман; пер. з англ. В.Я. Алтаєв. - М.: Світ, 1992. - 240 с.

9. Y-H Pao Адаптивне розпізнавання образів і нейронні мережі / Ян ХВУ Пао - Бостон: Addison-Wesley, 1989 - 164 с.

10. Ян Д.Є. Нова технологія розпізнавання символів. Теорія, практична реалізація, перспективи на майбутнє / Д.Є. Ян, К.В. Онисимович, А.Л. Шамис - М.: Препринт, 1995 - 180 с.

11. Щепин Є.В. До топологическому підходу до аналізу зображень. Геометрія, топологія та додатки (Міжвузівський збірник наукових статей) / Є.В. Щепин, Г. Непомнящий - М.: Світ, 1990 - 315 с.

12. А.В. Місюра використовує штучні нейронні мережі для розпізнавання рукописних символів / Олександр Місюра - М.: Едіторіал УРСС, 1998 - 142 с.

13. ABBYY FineReader [Електронний ресурс] - 2018. - Режим доступу до ресурсу: https://www.abbyy.com/en-ee/finereader/

14. Назустріч прогнозному [Електронний ресурс] - 2018. - Режим доступу до ресурсу: https://openreview.net/pdf?id=H1u8fMW0b

15. Комп'ютерна модель технологічної системи [Електронний ресурс] - 2018. - Режим доступу до ресурсу: https://web.posibnyky.vntu.edu.ua/fksa/2kvetnyj_komp'yuterne_modelyuvannya_system_procesiv/t2/24..htm

16. Кальман Р.Е. Новий підхід до завдань з лінійної фільтрацією і пророкуванням. Журнал інженерії рідин / Калман Р.Е. - Buffalo: ASME, 1960 - 50 с.

17. Баум Л. Статистичний висновок для імовірнісних функцій для кінцевих ланцюгів Маркова. Аннали математичної статистики / Леонард Е. Баум, Тед Петрі - Бічвуд: IMS, 1966 - 1630 с.

18. Принцип вільної енергії: теорія єдиного мозку [Електронний ресурс] - 2010. - Доступ до ресурсу: https://www.nature.com/articles/nrn2787

19. Сприйняття як гіпотези: мішки як експерименти. Межі в психології [Електронний ресурс] - 2012. - Режим доступу до ресурсу: https://www.frontiersin.org/articles/10.3389/fpsyg.2012.00151/full

20. Швидке виявлення особи з кількома видами. У Proc. комп'ютерного зору і розпізнавання образів. [Електронний ресурс] - 2003. - Режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.145.7598

21. Розрахунковий моделювання зорової уваги. Природні огляди нейробіології [Електронний ресурс] - 2001. - Місце для доступу до ресурсу: https://www.nature.com/articles/35058500

22. Впровадження системи кодування зображень з ямками для зменшення смуги пропускання зображення. In Human Vision and Electronic Imaging [Електронний ресурс] - 2001. - Режим доступу до ресурсу: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/2657/0000/Implementation-of-a-foveated - image -coding-system-for-image-bandwidth / 10.1117 / 12.238732.short? SSO = 1