НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. ІГОРЯ СІКОРСЬКОГО

факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету

кафедра обчислювальної техніки
(повна назва кафедри

До захисту допущено
Завідувач кафедри
Стіренко С.Г. _____
(прізвище, ініціали) (підпис)

“___”_____2019 р.

Дипломний проект
освітньо-кваліфікаційного рівня “бакалавр”
(назва ОКР)

з напряму підготовки (спеціальності) **6.050103 « Програмна інженерія»**
(код та назва напряму підготовки або спеціальності)

на тему «Перевірка унікальності графічних зображень »

Виконала: студентка 4 курсу, групи ІП-53
Литвиненко Анна Сергіївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асист. каф. ОТ Подрубайло Олександр Олександрович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант нормоконтроль д.т.н.,проф. Сімоненко В.П. _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент
_____
(підпис)

Київ - 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО"

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету

Обчислювальної техніки

Освітньо-кваліфікаційний рівень **бакалавр**

Напрям підготовки **6.050103 « Програмна інженерія »**

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____
(прізвище ініціали) (підпис)

"___"_____2019 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Литвиненко Анни Сергіївни _____
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Перевірка унікальності графічних зображень*
керівник проекту (роботи)_ _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "23"04 2019 року №1180-С
1. Термін здачі студентом закінченого проекту (роботи)_____ *2019*р.
3. Вихідні дані до проекту (роботи) *технічна документація*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Опис предметної області, дослідження методу перевірки унікальності графічних зображень з допомогою нейромереж та завантаження зображень до сховища
5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень)
схема алгоритму роботи програми, схема взаємодії класів при перевірці на унікальність зображення, UML діаграма класів програми.

6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	*Затвердження теми роботи*	*10.12.2019-15.12.2019*	
2.	*Вивчення та аналіз завдання*	*15.12.2019-15.03.2019*	
3.	*Розробка архітектури та загальної структури систем*	*15.03. 2019-25.03. 2019*	
4.	*Розробка структур окремих підсистем*	*25.03. 2019-5.04. 2019*	
5.	*Програмна реалізація системи*	*5.04. 2019-15.04. 2019*	
6.	*Оформлення пояснювальної записки*	*15.04. 2019-20.05. 2019*	
7.	*Захист програмного продукту*	*25.04. 2019*	
8.	*Передзахист*	*29.05. 2019*	
9.	*Захист*	*20.06. 2019*	

Студент-дипломник _____
(підпис)

Керівник роботи _____
(підпис)

Анотація

В бакалаврській роботі було запропоновано алгоритм перевірки унікальності графічних зображень під час завантаження їх до сховища.

Було створено програму, що реалізує запропонований алгоритм що дозволяє перевіряти на унікальність зображення під час їх завантаження до сховища. Це дозволяє прискорити роботу зі сховищем та уникнути дублікатів зображень. Програмний продукт реалізовано на мові Java.

Аннотация

В бакалаврской работе было предложено алгоритм проверки на уникальность графических изображений во время их загрузки в хранилище.

Была создана программа, реализующая предложенный алгоритм позволяющий проверять на уникальность изображения во время их загрузки в хранилище. Это позволяет ускорить работу с хранилищем и избежать дубликатов изображений. Программный продукт реализован на языке Java.

Annotation

In the bachelor’s work, an algorithm was proposed for checking the uniqueness of graphic images during their loading into the storage.

A program has been created implements the proposed algorithm that allows to check for uniqueness of the image during their loading into the storage. This allows to speed up work with the storage and avoid duplicate images. The software is implemented in Java.

Справки	Формат	*Позначення*	*Найменування*	Кільк. аркушів	№ екземпляра	*Примітки*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Перевірка унікальності	3		
			графічних зображень.			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Перевірка унікальності	52		
			графічних зображень.			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Перевірка унікальності	1		
			графічних зображень.			
			Схема алгоритму			
			роботи програми			
	А4	***ІАЛЦ.467200.005 Д2***	Перевірка унікальності	1		
			графічних зображень.			
			Схема взаємодії класів			
			при перевірці на			
			унікальність зображення			

					ІАЛЦ.467200.001 ОА			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Литвиненко			Перевірка унікальності графічних зображень. Опис альбому	Літ.	Арк.	Аркушів
Перев.		Подрубайло					1	2
						НТУУ "КПІ" ім. Ігоря Сікорського, ФІОТ, кафедра ОТ, гр. ІП-53		
Н. контр.		Сімоненко						
Затв.		Стіренко С.Г.						

Справки	Формат	***Позначення***	***Найменування***	Кільк. аркушів	№ екземпляра	***Примітки***
	А4	***ІАЛЦ.467200.006 Д3***	Перевірка унікальності	1		
			графічних зображень.			
			UML діаграма класів			
			програми			
	А4	***ІАЛЦ.467200.007 А1***	Перевірка унікальності	11		
			графічних зображень.			
			Текст програми			

					ІАЛЦ.467200.001 ОА	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

Технічне завдання
до дипломного проекту

на тему: «Перевірка унікальності графічних зображень»

Київ – 2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку програми по темі «Перевірка унікальності графічних зображень». Область застосування: оптимізація процесу завантаження зображень до сховищ методом перевірки унікальності з допомогою технології нейромереж.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи освітньо-кваліфікаційного рівня «бакалавр програмної інженерії», затверджене кафедрою спеціалізованих комп'ютерних систем Національного технічного Університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка моделі інструменту перевірки унікальності графічних зображень під час завантаження до сховищ.

4. ТЕХНІЧНІ ВИМОГИ

4.1. Вимоги до продукту, що розробляється

- Незалежність від платформи використання
- Зручність використання
- Висока швидкість роботи

4.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 98, MS Windows XP, MS Windows 7, MS Windows 8, Unix OS.

4.3. Вимоги до програмного забезпечення

- Комп’ютер на базі процесора Intel або AMD від 256 МГц
- Оперативної пам'яті не менше 512 Мбайт
- Вільний простір на диску не менше ніж 10 Мбайт

5. ЕТАПИ РОЗРОБКИ

Затвердження теми роботи	15.12.2018
Вивчення та аналіз завдання	15.03.2019
Розробка архітектури та загальної структури cucmeми	25.03. 2019
Розробка структур окремих підсистем	5.04. 2019
Програмна реалізація системи	15.04. 2019
Оформлення пояснювальної записки	20.05. 2019
Захист програмного продукту	25.05. 2019
Передзахист	29.05. 2019
Захист	20.06. 2019

Пояснювальна записка
до дипломного проекту

на тему: «Перевірка унікальності графічних зображень»

Київ – 2019

ЗМІСТ

ВСТУП

Інформаційні технології відіграють дуже важливу роль у сучасному світі. Кожен з нас використовує безліч електронних засобів кожного дня. Інформатизація у наш час торкається майже усіх сфер нашого життя, починаючи з повсякденних потреб у спілкуванні та, завершуючи, розвагами та науковою діяльністю.

Розглядаючи будь-яку сферу можна побачити, що чим більший розвиток – тим більші потреби. У сфері інформатизації однією з важливіших проблем можна вважати швидкість обробки інформації.

Інтернет ресурси вміщають у собі непомірну кількість такої інформації, як фото та відео. Через їх велику кількість постає проблема у обробці, сховищах та їх оптимізації. Чим більше інформації потрібно обробляти та зберігати, тим потужнішим повинен бути комп’ютер для її обробки та тим більшим повинно бути сховище.

Якщо говорити про збереження даних, то зараз існують багато варіантів розподілених сховищ та баз для великих об’ємів інформації, але постає проблема у обробці такої великої кількості та у часі такої обробки. Здебільшого величезні сховища для фото використовуються у так званих «фотобанках». Це ресурси для пошуку різноманітних фото, зображень та малюнків. Кожного дня до таких ресурсів завантажується дуже велика кількість матеріалу та існує вірогідність потрапляння дублікатів.

Постає питання у оптимізації збереження та обробки фото, щоб сховища таких ресурсів не були переповнені та обробка запитів була швидшою та менш ресурсозатратною.

Такі програми існують, але здебільшого вони призначені для локального очищення галереї комп’ютера і не для безкоштовного розповсюдження. Більшість з них працюють за стандартним алгоритмом, який має безліч недоліків та оцінка схожості зображень досить неточна.

Однією з сучасних технологій є нейромережі та з їх допомогою задачу пошуку схожих зображень можна разв’язати більш точно та оптимально.

В даній роботі розглянуті алгоритми для пошуку схожих зображень. Описано принципи роботи та реалізації нейромереж. Розглянуто механізм використання нейромережі для пошуку схожих зображень.

Метою цієї роботи є оптимізація обробки зображень під час завантаження їх до сховищ. Для цього розробляється алгоритм перевірки унікальності графічних зображень з допомогою нейромереж, спрямований на оптимізацію збереження зображень до сховищ.

РОЗДІЛ 1
ОСНОВНІ ВИЗНАЧЕННЯ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Цифрове зображення

Цифрове зображення - масив даних, отриманий шляхом дискретизації (аналого-цифрового перетворення) оригіналу. Після кодування за допомогою певного алгоритму і запису на носій, цей масив даних стає файлом.

У сучасному процесі поліграфічного виробництва всі ілюстрації й елементи оформлення представлені цифровими зображеннями різних типів. Цифрові зображення за способом дискретизації оригіналу поділяються на:

1) Растрові. До растрових зображень відносять двовимірні масиви даних (матриці пікселів), кожен елемент яких представляє ділянку оригіналу з усередненим колірним показником. Основні характеристики растрового зображення - розмір та глибина кольору. Розмір зображення в пікселях - це кількість рядків і стовпців матриці, що використовуються для зберігання зображення. Розмір цифрового зображення можна довільно змінювати, змінюючи фізичний розмір картинки при друку, при цьому розмір матриці пікселів буде залишатися незмінним. Глибина кольору - це характеристика, яка визначає якість відтворення кольору, кількість відтінків, які можуть відображати елементи матриці пікселів.
2) Векторні. Іншим видом цифрових зображень є векторні зображення. Найменшими елементами векторного зображення є вектор і крива Безьє. Вектор в комп'ютерній графіці — це відрізок, що з'єднує дві точки з заданими координатами. Основним керувальним елементом кривої Безьє є вузол, також так звана контрольна точка або контрольна вершина. Ступінь кривини лінії визначається координатами вузла і двох керувальних точок.Контур елемента векторного зображення в цифровому вигляді являє собою масив даних, що містить координати

контрольних та керувальних точок, а також характеристики кривої в цілому — її товщину, колір, напрямок, а якщо крива замкнута — то й колір і тип заливки.Примітиви являють собою прості геометричні форми, які в масиві даних кодуються цілком, без поділу на криві Безьє і вектори, умовним кодом тієї чи іншої геометричної фігури, а також кодами розміру фігури, її координатами, кодами типу і кольору заливки фігури, товщини і кольору контуру та інших характеристик.

3) Змішаного типу. Цифрові зображення змішаного типу являють собою масиви даних, що містять інформацію як у вигляді матриці пікселів, так і у вигляді опису векторів, кривих Безьє, примітивів і текстових блоків. В основі вертикальної структури векторно-растрових зображень лежить поняття шару. Шар - це область даних, що містить інформацію про окремий елемент вертикальної структури зображення. Векторно-растрові зображення отримують з вихідних векторних і растрових елементів шляхом зведення за допомогою графічних редакторів. Також умовно до зображень змішаного типу слід віднести результати роботи програм комп'ютерної верстки, в яких основними векторними елементами виступають текстові блоки. Зображення змішаного типу поєднують в собі переваги й недоліки тих типів зображень, які присутні в них у вигляді елементів (шарів).

1.2 Штучна нейронна мережа

Штучна нейронна мережа — це мережа простих елементів, званих нейронами, які отримують вхід, змінюють свій внутрішній стан (збудження) відповідно до цього входу, і виробляють вихід, залежний від входу та збудження. Мережа утворюється з'єднанням виходів певних нейронів зі входами інших нейронів з утворенням орієнтованого зваженого графу. Ваги, як і функції, що обчислюють збудження, можуть змінюватися процесом, званим навчанням, який керується правилом навчання[1].

Складовими неронної мережі є нейрони, з'єднання та ваги, функції поширення та правила навчання.

Нейрон з міткою j, що отримує вхід $p_j(t)$ від нейронів-попередників, складається з наступних складових:

1) збудження (англ. activation) $a_j(t)$, що залежить від дискретного параметра часу,
2) порогу (англ. threshold) θ_j, що залишається незмінним, якщо його не змінить функція навчання,
3) функції збудження (англ. activation function) f, яка обчислює нове збудження в заданий час $t+1$ з $a_j(t)$, θ_j та мережевого входу $p_j(t)$, даючи в результаті відношення $a_j(t+1) = f(a_j(t), p_j(t), \theta_j)$,
4) функції виходу (англ. output function) f_{out}, яка обчислює вихід з активації $o_j(t) = f_{out}(a_j(t))$.

Функція виходу часто є просто тотожною функцією.

Нейрон входу (англ. input neuron) не має попередників, а слугує інтерфейсом входу для всієї мережі. Аналогічно, нейрон виходу (англ. output neuron) не має наступників, і відтак слугує інтерфейсом виходу для всієї мережі.

Мережа складається зі з'єднань, кожне з яких передає вихід нейрону i до входу нейрону j. В цьому сенсі i є попередником j, а j є наступником i. Кожному з'єднанню призначено вагу w_{ij}.

Функція поширення обчислює вхід $p_j(t)$ до нейрону j з виходів $o_j(t)$ нейронів-попередників, і зазвичай має вигляд $p_j(t) = \sum_i o_j(t) w_{ij}$.

Правило навчання — це правило або алгоритм, який змінює параметри нейронної мережі, щоби заданий вхід до мережі видавав придатний вихід. Цей процес навчання зазвичай полягає в зміні ваг та порогів змінних мережі.

1.3 Згорткова нейронна мережа

Згоорткові нейроонні мережі в машинному навчанні(ЗНМ) — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовується до аналізу візуальних зображень.

ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоб вимагати використання мінімального обсягу попередньої обробки. Вони відомі також як інваріантні відносно зсуву або просторово інваріантні штучні нейронні мережі, виходячи з їхньої архітектури спільних ваг та характеристик інваріантності відносно паралельного перенесення.

Згорткові мережі було натхнено біологічними процесами, в яких схему з'єднання нейронів натхнено організацією зорової кори тварин. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле.

ЗНМ використовують порівняно мало попередньої обробки, в порівнянні з іншими алгоритмами класифікування зображень. Це означає, що мережа навчається обробці зображень, що в традиційних алгоритмах розроблялися вручну. Ця незалежність у конструюванні ознак від апріорних знань та людських зусиль є великою перевагою.

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації.

1.3.1 Архітектура та принципи роботи

У звичайному перцептроні, який представляє собою повнозв’язну нейронну мережу, кожен нейрон пов’язаний з усіма нейронами попереднього шару, причому кожен зв’язок має свій персональниій ваговий коефіцієнт(рис. 1.1).

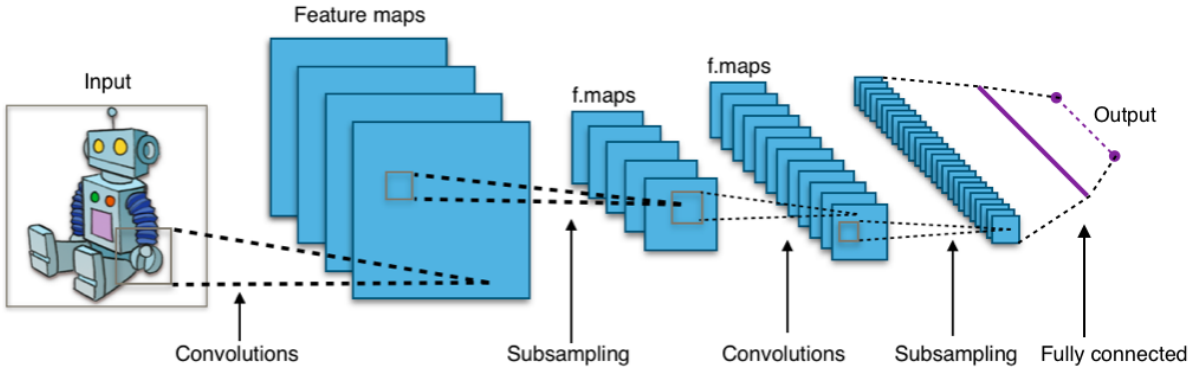


Рис. 1.1 Архітектура згорткової нейронної мережі[12]

У згортковій нейронній мережі в операції згортки використовується лише обмежена матриці ваг невеликого розміру, яку «рухають» по всьому оброблюваному шару, формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Тобто для різних нейронів вихідного шару використовується одна і та ж матриця ваг, яку також називають ядром згортки. Її інтерпретують як графічне кодування якої-небудь ознаки, накприклад, наявність похилої лінії під певним кутом. Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак. Зрозуміло, що у згортковій нейронній мережі набір ваг не один, а набір, що кодує елементи зображення(наприклад лінії і дуги під різними кутами). При цьому ядра згортки не закладаються дослідником зазадлегідь, а формуються самостійно шляхом навчання мережі класичним методом зворотнього поширення помилки. Прохід кожним набором ваг формує свій власний примірник карти ознак, роблячи нейронну мережу багатоканальною(багато незалежних карт ознак на одному шарі). Також слід зазанчити, що при переборі шару матрицею ваг її пересувають зазвичай не на певний крок(розмір матриці), а на невелику відстань. Так, наприклад, при розмірності матриці ваг 5*5 її переміщають на один або на два нейрона(пікселя) замість п’яти, щоб не «переступити» шукану ознаку.

Операція субдискретизації виконує зменшення розмірності сформованих карт ознак. У даній архітектурі мережі вважається, що інформація про факт наявності шуканої ознаки важливіше точного знання його координат, тому з кількох сусідніх неронів карти ознак вибирається максимальній і приймається за один нейрон цщільненої карти ознак меншої розмірності. За рахунок цієї операції, крім прискорення подальших обчислень, мережа стає більш інваріантною до масштабу вхідного зображення.

Розглядаючи типову структуру згорткової нейронної мережі детальніше, можна побачити велику кількість шарів. Після початкового шару(вхідного зображення) сигнал проходить серію згорткових шарів, в яких чергується згортка та субдискретизація. Чергування шарів дозволяє складати «карти ознак» з картою знак, на кожному наступному шарі карта зменшується в розмірі, але не збільшується кількість каналів.

На практиці це означає здатність розпізнавання складних ієрархій ознак. Зазвичай після проходження декількох шарів карта ознак вироджується в вектор або навіть скаляр, але таких карт ознак стають сотні. На виході згорткових шарів мережі додатково встановлюють кілька шарів повної нейронної мережі (перцептрон), на вхід якого подаються кінцеві карти ознак.

1.3.2 Згортковий шар

Шар згортки - це основний блок згорткової нейронної мережі. Шар згортки включає в себе для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати матричного добутку для кожного фрагмента). Вагові коефіцієнти ядра згортки (невеликої матриці) невідомі і встановлюються в процесі навчання.

Особливістю зготкового шару є порівняно невелика кількість параметрів, яке встановлюється при навчанні. Так наприклад, якщо вихідне

зображення має розмірність 100 × 100 пікселів по трьом каналам (це значить 30000 вхідних нейронів), а згортковий шар використовує фільтри з ядром 3x3 пікселя з виходом на 6 каналів, тоді в процесі навчання визначається тільки 9 ваг ядра, однак по всім сполученням каналів, тобто 9 × 3 × 6 = 162, в такому випадку даний шар вимагає знаходження тільки 162 параметрів, що істотно менше кількості шуканих параметрів повно нейронної мережі.

1.3.3 Шар активації

Скалярний результат кожної згортки потрапляє на функцію активації, яка представляє собою будь-яку нелінійну функцію. Шар активації зазвичай логічно пов'язують з шаром згортки (вважають, що функція активації вбудована в шар згортки). Функція нелінійності може бути будь-якою за бажанням дослідника, традиційно для цього використовували функції типу гіперболічного тангенса ($f(x) = \tanh(x), f(x) = |\tanh(x)|$) або сигмоіди ($f(x) = (1 + e^{-x})^{-1}$). Однак в 2000х роках була запропонована і досліджена нова функція активації - ReLU (скорочення від англ. Rectified linear unit), яка дозволила суттєво прискорити процес навчання і одночасно спростити обчислення (за рахунок простоти самої функції), що означає блок лінійної ректифікації, що обчислює функцію $f(x) = \max(0, x)$. Тобто по суті це операція відсікання негативній частині скалярної величини. Станом на 2017 рік ця функція і її модифікації (Noisy ReLU, Leaky ReLU і інші) є найбільш часто використовуваними функціями активації в глибоких нейромережах, зокрема, в згорткових. Існує методика визначення оптимального числа блоків лінійної ректифікації.

1.3.4 Шар субдискретизації

Шар пулінгу (інакше підвибірки, субдіскретізації) являє собою нелінійне ущільнення карти ознак, при цьому група пікселів (зазвичай розміру 2 × 2) ущільнюється до одного пікселя, проходячи нелінійне

перетворення. Найбільш використовувана при цьому функція максимуму. Перетворення зачіпають непересічні прямокутники або квадрати, кожен з яких скорочується в один піксель, при цьому вибирається піксель, що має максимальне значення. Операція пулінгу дозволяє істотно зменшити просторовий обсяг зображення. Пулінг інтерпретується так: якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки дуже детальне зображення вже не потрібно, і воно ущільнюється до менш детального. До того ж фільтрація вже непотрібних деталей допомагає не перенавчати нейромережу. Шар пулінгу, як правило, вставляється після шару згортки перед шаром наступної згортки.

Крім пулінгу з функцією максимуму можна використовувати і інші функції - наприклад, середнього значення або L2-нормування. Однак практика показала переваги саме пулінгу з функцією максимуму, який включається в типові системи.

З метою більш агресивного зменшення розміру одержуваного, все частіше знаходять поширення ідеї використання менших фільтрів або повна відмова від шарів пулінгу.

1.3.5 Переваги згорткової нейронної мережі

1) Один з найкращих алгоритмів з розпізнавання та класифікації зображень.
2) У порівнянні з повнозв’язною нейронною мережею (типу перцептрона) - набагато менша кількість настроюваних ваг, так як одне ядро ваг використовується повністю для всього зображення, замість того, щоб робити для кожного пікселя вхідного зображення свої персональні вагові коефіцієнти. Це підштовхує нейромережу при навчанні до узагальнення демонстрованої інформації, а не попіксельного запам'ятовування кожної показаної картинки в міріадах вагових коефіцієнтів, як це робить перцептрон.

3) Зручне розпаралелювання обчислень, а отже, можливість реалізації алгоритмів роботи і навчання мережі на графічних процесорах
4) Відносна стійкість до повороту та зміщення зображення, що розпізнається.
5) Навчання за допомогою класичного методу зворотнього поширення помилки.

1.4 Аналіз існуючих рішень

У наш час поширена проблема пошуку нечітких дублікатів у колекції зображень, їх кластеризації та розпізнавання образів на зображеннях. Деякі з алгоритмів, для вирішення цих задач також можуть бути частково використані у розв'язанні задачі визначення дублікатів зображень при завантаженні до сховища.

1.4.1 Tiny Images

Було доведено, що для розпізнавання людиною чорно-білого зображення, розмір його повинен бути не меньше 64*64 пікселі, а для розпізнавання кольорового зображення з точністю 80% і більше потрібно всього 32*32 пікселі[6]. Алгоритм TinyImages базується на створенні бази зображень розміром 32*32 у якій ми порівнюємо цільове зображення з іншими на предмет схожості[3].

Першим кроком обчислюється сума квадратів різниць(SSD) для порівняння зображень. Пізніше будуть розглянуті зміни для підвищення якості та можливості порівняння зображень з переворотом та масштабуванням. SSD розраховується для двох зображень I_1 та I_2 (нормалізовані) за такою формулою (1.1):

$$D_{ssd}^2 = \sum_{x,y,c} (I_1(x,y,c) - I_2(x,y,c))^2 \quad (1.1)$$

Нормалізація для кожного зображення відбувається шляхом перетворення зображення у вектор, що поєднує три кольорових канала. Нормалізація не змінює кольори зображення, тільки загальну яскравість. $1/f^2$ властивість спектру потужності природних образів означає, що відстань між двома зображеннями можна апроксимувати за допомогою декількох основних компонент. Ми обчислюємо відстань $D_{ssd}^2 = 2 - 2\sum_{n=1}^{C} v_1(n)v_2(n)$, де $v_i(n)$ це n-тий коефіцієнт основної складової для і-го зображення (нормованого так, що $\sum_n v_i(n)^2 = 1$), та C - кількість компонент, що використовується для наближення відстані. Визначимо S_N, як набір з N точних найближчих сусідів та S_M, як набір з M приблизних найближчих сусідів.

Можна підвищити ефективність розпізнавання за допомогою більш вдосконалених вимірів схожості зображень. Вводяться дві додаткові міри подібності між парою нормалізованих зображень I_1 та I_2, тобто включити інваріанти до просторових перетворень.

Для того, щоб включити інваріантність до малих переворотів, масштабування та віддзеркалення, визначається міра подібності (1.2):

$$D_{warp}^2 = \min_{\theta} \sum_{x,y,c} (I_2(x,y,c) - T_\theta[I_2(x,y,c)])^2 \qquad (1.2)$$

У цьому виразі мінімізується схожість, трансформуючи I2(горизонтальне віддзеркалення, перевороти та масштабування до 10 зміщень пікселів). Параметр трансформації θ оптимізується градієнтним спуском.

Дозволяються додаткові спотворення зображень через зміщення кожного пікселя окремо в межах 5 на 5 вікна для зменшення суми квадратів різниць. У цьому випадку мінімум може бути знайдений за допомогою вичерпної

оцінки всіх зміщень, яка можлива тільки при низькому розширенні зображень.

$$D_{shift}^2 = \min_{|D_{x,y}| \le w} \sum_{x,y,c} (I_1(x,y,c) - \widehat{I_2}\,(x + D_{x,y} + D_{y,c}))^2 \qquad (1.3)$$

Для того, щоб отримати найточніші результати, I_2 (1.4) ініціалізується параметрами викривлення, отриманими після оптимізації D_{warp}.

$$\widehat{I_2} = T_\theta[I_2]. \qquad (1.4)$$

На рис. 1.2 та рис. 1.3 показані декілька зображень, які підбиралися з допомогою метрик та показані результуючі «сусіди» трансформовані у відповідності з оптимальними параметрами, що мінімізують кожну міру подібності. Фото показують два кандидати у сусіди: один відповідає цільовій семантичній категорії, а другий відповідає немправильному співпадінню. Для D_{warp} (1.2) та D_{shift} (1.3) показано найбільш наближені зображення до цільового. D_{warp} найкраще транслює, масштабує та горизонтально віддзеркалює зображення кандидата у відповідності з цільовим зображенням. D_{shift} додатково оптимізує викривлення, яке відбувається у результаті використання D_{warp}, який дозволяє пікселям рухатися для того, щоб мінімізувати відстань до цільового зображення.

Рис. 1.2 Знаходження схожих зображень з допомогою метрик[3]

Рис. 1.3 Знаходження схожих зображень з допомогою метрик[3]

1.4.2 Метод пошуку зображень за прикладом

Для вирішення задач пошуку зображень колекції за прикладом використовується алгоритм, який заснований на тому, що має місце зображення у вигляді нечіткої кольорової гістограми[5].

Метод розрахунку нечіткої кольорової гістограми базується на методі, що включає наступні етапи:

1) Приведення вихідного зображення до розміру, яке не перебільшує 100 × 100 точок. При цьому проводиться збереження пропорцій вихідного зображення і бікубічна інтерполяція кольорів.
2) Приведення отриманого зображення до універсального кольорового простору CIE L * a * b * [2].
3) Розрахунок значень функції приналежності у кожній точці зображення.
4) Побудова гістограми значень функції приналежності.

За оригінальним методом розглядаються роздільні функції приналежності для кожного кольорового каналу, форми яких ілюстровано на рис. 1.4 – 1.6.

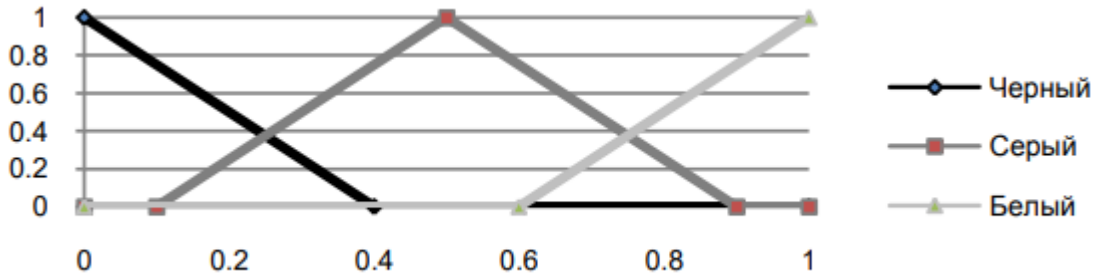


Рис. 1.4 Функції приналежності канал *L**

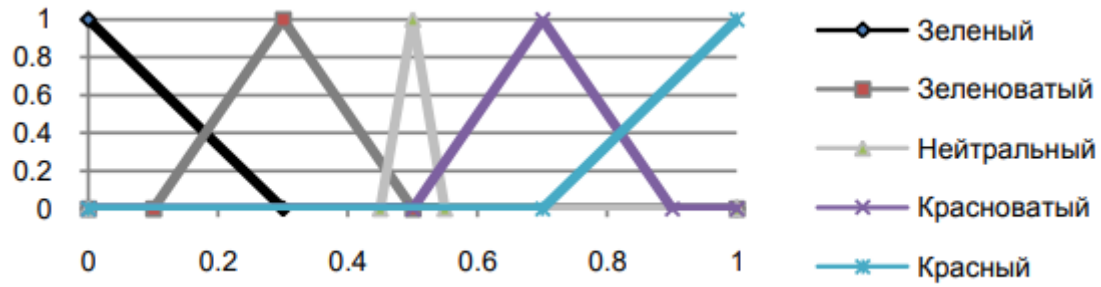


Рис. 1.5 Функції приналежності канал *a**

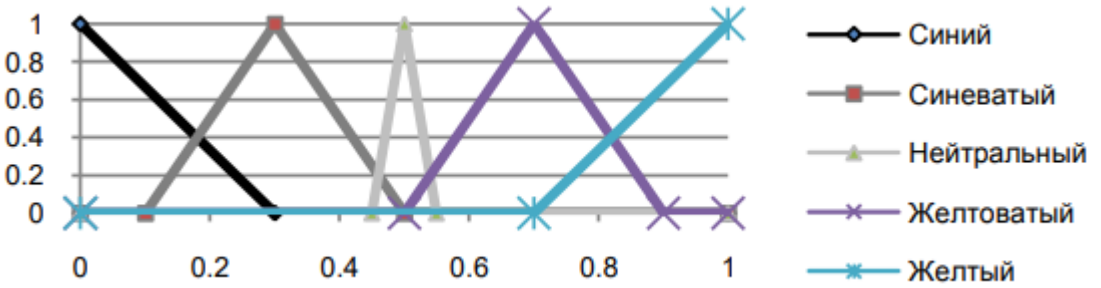


Рис. 1.6 Функції приналежності канал *b**

Основною ввідмінністю цього методу являється те, що гістограми будуються без використання процедури дефазифікації. У кожній точці зображення будується вектор 75-вимірного простору, компоненти якого являються середнім арифметичним значень трьох функцій приналежності. Після чього виконується усереднення даних векторів.

Відстань між схожими векторами, які описують зображення, розраховується на основі функції перетину гістограм (1.5):

$$d(P,Q) = 1 - \frac{\sum_{i=1}^{75} min(P_i, Q_i)}{min(\sum_{i=1}^{75} P_i, \sum_{i=1}^{75} Q_i)} \quad (1.5)$$

де P, Q – порівнювані вектори.

На відміну від базового методу, у рамках якого для зображення вираховується єдина гістограма, у цьому методі пропонується рахувати

шість додаткових гістограм, які будуються для ділянок зображення, показаних на рис. 1.7. Це дозволяє вираховувати відстань між зображеннями з урахуванням геометричних перетворень, таких як відображення.

Рис. 1.7 Ділянки зображення, які використовуються для побудови додаткових колорових гістограм[5]

Відстань між зображеннями *P*, *Q*, заданими гістограмами $(P^0, \dots, P^6)$ та $(Q^0, \dots, Q^6)$, де індексом «0» позначена гістограма, яка побудована за повним зображенням, розраховується як зважена середня відстань між гістограмами(1.6):

$$D(P,Q) = \sum_{i=0}^{6}\sum_{i=0}^{6} d\left(P^i, Q^j\right) * \delta_{ij} \tag{1.6}$$

де δ_{ij} – елемент матриці, який визначає вагу відстані між гістограмами з індексами *i* та *j*. Дана матриця була задана у вигляді(1.7):

$$\Delta = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (1.7)$$

Ранжування зображень виконувалось у порядку зростання від зображення-запиту до зображення-кандидату.

ВИСНОВКИ ДО РОДЗІЛУ 1

Нейромережі, а особливо згорткові, являються сучасною та перспективною технологією, яка вже активно використовується у сьогоденні. Згорткові нейромережі набули популярності через стійкість алгоритму до модифікацій зображення, узагальнення характеристик зображення, та значне прискорення у обчисленнях, через можливість розпаралелювання. Всі ці характеристики дають перевагу у якості та швидкості розпізнавання.

Основною проблемою розпізнавання дублікатів зображень являється швидкість обчислень та точність розпізнавання. Розглянуті рішення можуть мати місце у вирішенні цієї задачі, але будуть істотно програвати за цими показниками. Отже, для вирішення поставленої задачі потрібно реалізувати алгоритм на базі згорткової нейромережі.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ ТА РОЗРОБКА МОДЕЛІ

2.1 Вибір інструментів розробки

На даний момент існує величезний вибір іструментів для розробки програмного забезпечення.

Також існує безліч мов програмування. Найпопулярнішими являються: Java, C#, Go, Java Script, Ruby та інші. Кожна мова має свої особливості, недоліки та переваги. Головними критеріями при виборі мови є: швидкість, надійність, простота використання та кросплатформеність. Мова Java має усі з перелічених характеристик, тому була обрана для вирішення задачі диплому.

Після вибору мови програмування потрібно обрати середовище розробки(IDE). Зараз найпопулярнішими IDE, для роботи з мовою Java є: Intelij Idea, Eclipse та NetBeans.

Intellij Idea – комерційне інтегроване середовище розробки для розробки на різних мовах програмування від компанії JetBrains. Існує безкоштовна версія цього середовища розробки “Community Edition”. Community версія підтримує інструменти для проведення тестування у вигляді TestNG та JUnit, системи контролю версій CSV, Subversion, Mercurial та Git, засоби складання Maven, Ant, Gradle, мови програмування Java, Scala, Cloujure, Groovy та інші. Підтримується розробка мобільних додатків, для мобільної платформи Android. До складу входить модуль візуального програмування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з інших середовищ розробки. Доступні засоби інтеграції з системами відстеження помилок Jira, Trac, Redmine, Pivotal Tracker, GitHub та іншими. Також інсує платна версія «Ultimate Edition», яка розповсюджується безкоштовно у

навчальних цілях. Вона має у собі також можливість роботи з базами даних, таких як MySQL, PostgreSQL та інші. Існує можливість згенерувати діаграмми класів та компонентів програми. В цілому, Ultimate версія повністю задовольняє потреби при WEB-розробці.

Eclipse – вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для розробників на мові Java, засоби для управління сирцевими кодами, візуальні побудовники GUI тощо. Написаний в основному на Java, може бути використаний для розробки на Java, а за допомогою різних плагінів і на інших мовах програмування, включаючи Ada, C, C++. COBOL. Perl, PHP та інші. Підтримує велику кількість популярних нині технологій. Взаємодіє із системами контролю версій. Різні розширення дають змогу використання нових технологій накшталт Spring, як інтегрованого інструменту.

Net Beans – вільне інтегроване середовище розробки для мов програмування Java, JavaFX, C, C++, PHP, тощо. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Поширюється у сирцевих текстах під ліцензіями GPLv2 і CDDL. Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems, нині Oracle. За якістю і можливостями останні версії замагаються з найкращими інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду та інше. Net Beans підтримує плагіни, дозволяючи розробникам розширювати можливості середовища.

Після огляду середовищ розробки, їх переваг та недоліків, було вирішено обрати Intelij Idea через можливість роботи з базою, швидке та інтуїтивне доповнення коду та автоматичне коригування, через повну підтримку усіх технологій для WEB-розробки.

2.2 Вибір допоміжних технологій

Для того, щоб максимально повно вирішити поставлену задачу, знадобляться додаткові технології, використання баз даних тощо. Так, як розпізнавання зображень при завантаженні до сховища здебільшого може мати місце при роботі з WEB-ресурсами, тому було вирішено розробити WEB додаток.

Існують декілька технологій для створення WEB додатків на мові програмування Java. Одна з доволі «зрілих» технологій – Java EE(Java Enterprise Edition). Вона вже не популярна у сучасному світі, тому її на зміну прийшов фреймворк Spring.

Spring – найпопулярніший фреймворк для створення веб-додатків на Java. Spring складається з набору заздалегідь розроблених інтерфейсів, классів та зв'язків між ними. Модна також сказати, що Spring – це скоріш не конкретний фреймворк, а назва для цілого ряду невеликих фреймворків, кожен з яких виконує якусь роботу. У нього модульна структура. Це добре видно на рис. 2.1.

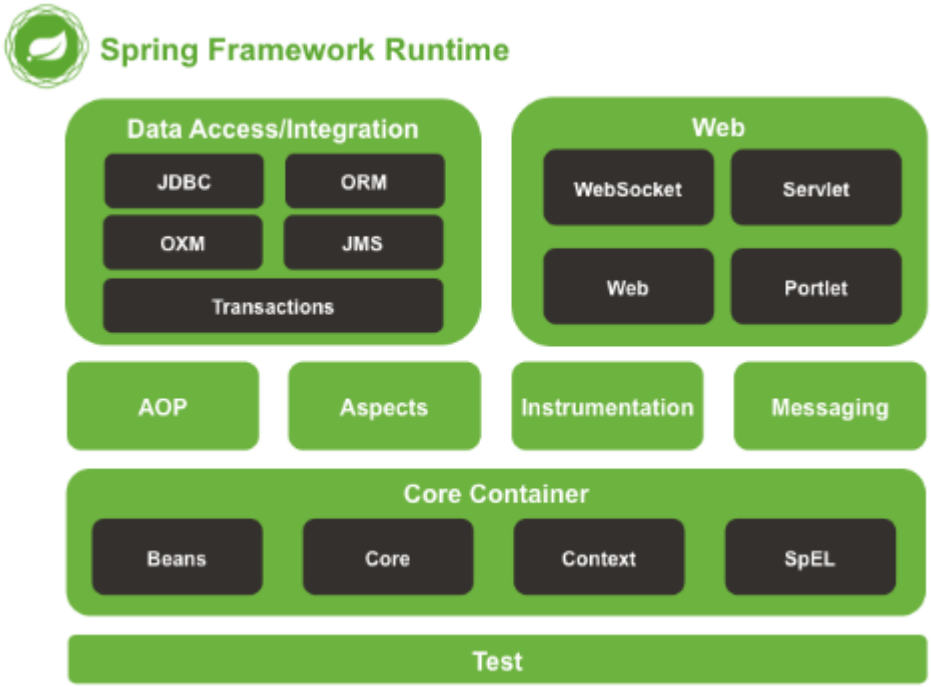


Рис. 2.1 Структура фреймворка Spring[13]

Модульність дозволяє підключати тільки ті модулі, що потрібні для розробки конкретного додатку.

Для роботи з базою даних буде використовуватись Spring Data JPA – це додатковий механізм для взаємодії з сутностями бази даних, для організації їх у репозиторії, витяг даних, їх зміна. Простота в тому, що здебільшого для проведення всіх цих операцій достатньо оголосити інтерфейс та його методи без імплементації.

Для більшого комфорту використання фреймворку Spring розробники створили наступний рівень – Spring Boot. Spring Boot – це проект на рівні виконання (IO Execution) IO Spring Framework, який дозволяє спростити настрювання Spring та підключення до нього модулів.

Переваги Spring Boot:

1) Легко використовується для розробки додатку на основі Spring с Java або Groovy Spring
2) Мінімізується час розробки та піднімає швидкодію
3) Уникає написання багатьох кодів прототипів, анотацій та конфігурацій XML
4) Легко дозволяє взаємодіяти з додатками Spring Boot з екологічними системами Spring таких, як Spring JDBC, Spring ORM, Spring Data, Spring Security та інші.
5) Слідує підходу «Принципи конфігурації за замовчуванням», щоб мінімізувати час та ресурси, вкладені у розробку додатку
6) Забезпечує вбудований сервер такий, як Tomcat, Jetty, щоб швидко та легко розроблювати та тестувати веб-додатки
7) Надає інструменти CLI(Command Line Interface) для розробки та тестування додатків Spring Boot з командного рядка легко та швидко
8) Забезпечує багато плагінів для швидкої розробки та тестування додатку Spring Boot, використовуючи інструменти Build такі, як Maven та Gradle.

9) Пропонує багато плагінів для легкої роботи з контейнерами, вбудованими базами даних та базами, які зберігаються у пам’яті

Для взаємодії з веб-додатком потрібно обрати її тип. Існують два найпопулярніших типа REST та SOAP.

REST(Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого додатку у мережі. Компоненти REST взаємодіють, як клієнти та сервери у мережі Інтернет. Для спілкування клієнта з сервером клієнту потрібно відправити запит. За REST запит повинен бути направленим на отримання даних, або модифікацію даних на сервері. У загальному запит повинен мати[8]:

1) HTTP дію, яка описує операцію, яка повинна виконатися на сервері(«POST», «GET», «DELETE», «UPDATE»)
2) Заголовок, який дозволяє клієнту передати інформацію про запит
3) Шлях до ресурсу
4) Опціонально може мати тіло, яке містить дані

Після отримання сервером запиту, він повинен обробити його та надіслати відповідь, яка повинна містити статус виконання. Статус має свій код і описує вдале виконання запиту, або ж помилку, яка могла статися.

Переваги REST:

1) Простота уніфікованого інтерфейсу
2) Відкритість компонентів до можливих змін для задоволення потреб
3) Прозорість зв’язків між компонентами системи для сервісних служб
4) Надійність(за рахунок відсутності необхідності зберігати інформацію про стан клієнта, яка може бути загублена)
5) Швидкодія(за рахунок використання кешу)
6) Простота інтерфейсів
7) Масштабованість

SOAP(Simple Object Access Protocol) – протокол обміну структурованими повідомленнями у розподіленому середовищі. Протокол використовується для обміну довільними повідомленнями у форматі XML. Може використовуватися з будь-яким протоколом прикладного рівня: FTP, HTTP, HTTPS та інші. На відміну від REST, у SOAP відправка повідомлень виконується тільки методом POST. SOAP використовує інтерфейси, які базуються на об’єктах і методах. Інтерфейси у SOAP можуть мати нелімітовану кількість методів, у той же час REST обмежений чотирма операціями.

Переваги SOAP:

1) Більш гнучкий тип даних
2) Підтримка заголовків та розширень

Недоліки SOAP:

1) Використання SOAP для передачі повідомлень збільшує їх об’єм та знижає швидкість обробки. У системах, де важлива швидкість частіш за все використовується передача XML документів через HTTP напряму, де параметри запиту передаються як звичайні HTTP параметри

Розглянувши технології, що можуть використовуватися при розробці та можуть робити її швидшою та комфортнішою, було обрано Spring Boot через простоту настроювання середовища та модулів та REST через швидкодію, простоту та можливість визначення статусу запиту.

2.3 Вибір фреймворка для роботи з нейромережею

Для роботи з нейромережами на мові Java існує декілька базових фреймворків. Основні з них це Neuroph та Deeplearning4j.

Neuroph - це об'єктно-орієнтована структура штучної нейронної мережі, написана на Java. Він може бути використаний для створення та навчання нейронних мереж у програмах Java. Neuroph надає бібліотеку класів

Java, а також інструмент GUI easyNeurons для створення та навчання нейронних мереж.

Основні класи Neuroph відповідають основним концепціям нейронної мережі, таким як штучний нейрон, шар нейрона, зв'язок нейрона, вага, передатна функція, функція введення, правило навчання тощо. Neuroph підтримує популярні архітектури нейронних мереж, такі як багатошаровий перцептрон з методом зворотнього поширення, мережі Кохонана та Хопфілда. Всі ці класи можна розширити і налаштувати для створення користувацьких нейронних мереж і правил навчання. Neuroph має вбудовану підтримку розпізнавання зображень[9].

Deeplearning4j – це бібілотека для машинного навчання, написана на Java і має обчислювальну базу з широкою підтримкою алгоритмів машинного навчання. Окрім базового набору реалізації, таких як згорткові нейромережі та інше, Deeplearning4j включає у себе реалізацію обмеженої машини Больцмана, мережу переконань, автокодер, автокодер з багатошаровим шумопоглинанням та рекурсивну мережу нейронних тензорів, word2vec, doc2vec і GloVe. Всі ці алгоритми включають розподілені паралельні версії, які інтегруються з Apache Hadoop і Spark [10].

Deeplearning4j може працювати разом з Tensorflow та Keras. Deeplearning4j має можливість імпортувати моделі з Tensorflow та інших Python фреймворків, якщо вони створені з допомогою Keras.

Ознайомившись з двома основними фреймворками для роботи з нейромережами було вирішено використати Deeplearning4j через більш широкий спектр реалізацій та можливу інтеграцію з Tensorflow.

2.4 Вибір архітектури згорткової нейронної мережі

Для того, щоб вирішити задачу розпізнавання дублікатів зображень, можна використати вже готову архітектуру згорткової нейромережі, через яку буде проходити вхідне зображення для отримання його характеристики.

2.4.1 LeNet-5

LeNet-5[14] використовувався у великих масштабах для автоматичної класифікації рукописних цифр банківських чеків у Сполучених Штатах. Ця мережа є згортковою нейронною мережею (CNN). CNNs є основою сучасного сучасного сучасного комп'ютерного зору на основі глибинного навчання. Ці мережі побудовані на трьох основних ідеях: локальних рецептивних полях, загальних вагах і просторовій підсистемі. Локальні рецептивні поля з загальними вагами є сутністю згорткового шару, і більшість описаних нижче архітектур використовують згорткові шари в тій чи іншій формі[11].

Ще однією причиною, чому LeNet є важливою архітектурою, є те, що до того, як вона була винайдена, розпізнавання символів було зроблено в основному за допомогою інженерної техніки вручну, за якою слідувала модель машинного навчання, щоб навчитися класифікувати спеціально розроблені функції. LeNet зробив ручні інженерні функції надлишковими, оскільки мережа автоматично вивчає найкраще внутрішнє представлення із сирих зображень.

За сучасними стандартами LeNet-5 - це дуже проста мережа. Він має лише 7 шарів, серед яких є 3 згорткові шари (C1, C3 і C5), 2 шари підсимплінгу (об'єднання) (S2 і S4) і 1 повнозв’язний шар (F6), за якими слідує вихідний шар. Згорткові шари використовують 5 на 5 згортань з кроком 1. Суб-дискретизуючі шари складають 2 на 2 середніх шари об'єднання. Сигмоподібні активації Tanh використовуються по всій мережі.

Перший шар. Вхід для LeNet-5 являє собою 32 × 32 зображення у градаціях сірого, що проходить через перший згортковий шар з 6 мапами характеристик або фільтрами розміром 5 × 5 та кроком 1. Розмір зображення змінюється від 32x32x1 до 28x28x6 (рис. 2.2).

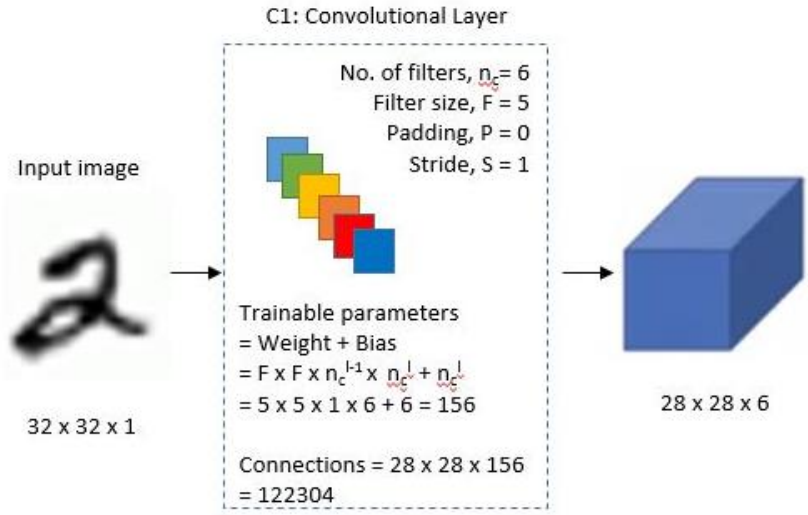


Рис. 2.2 Структура першого шару[14]

Другий шар. Потім LeNet-5 застосовує середній шар об'єднання або подсамплифицирующий шар з розміром фільтра 2 × 2 і кроком у два. Отримані розміри зображення будуть зменшені до 14x14x6 (рис. 2.3).

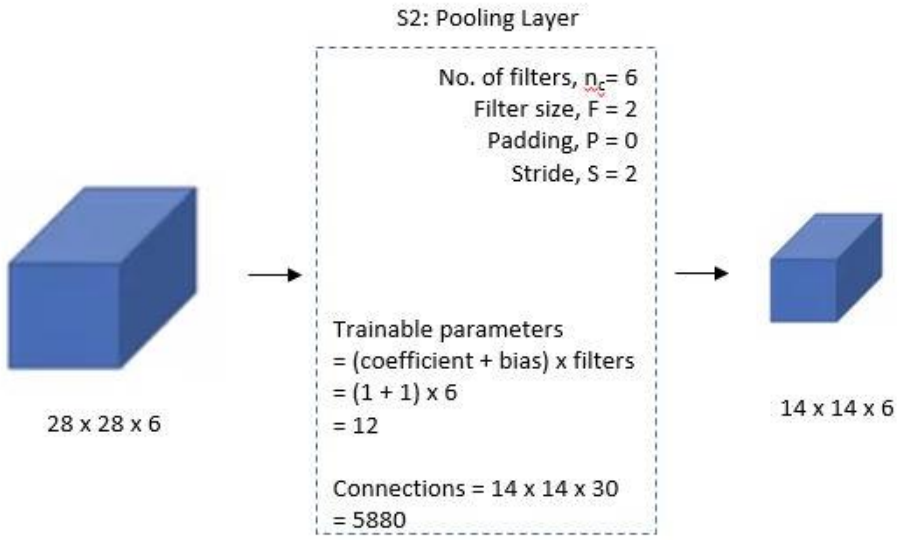


Рис. 2.3 Структура другого шару[14]

Третій шар. Далі є другий згортковий шар з 16 картами характеристик розміром 5 × 5 і кроком 1. У цьому шарі тільки 10 з 16 карт характеристик з'єднані з 6 мапами попереднього шару, як показано на рис. 2.4.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Рис. 2.4 З'єднання карт характеристик[14]

Основною причиною є порушення симетрії в мережі і збереження кількості з'єднань в розумних межах. Тому кількість навчальних параметрів у цьому шарі становить 1516 замість 2400, а також кількість з’єднань становить 151600 замість 240000 (рис. 2.5).

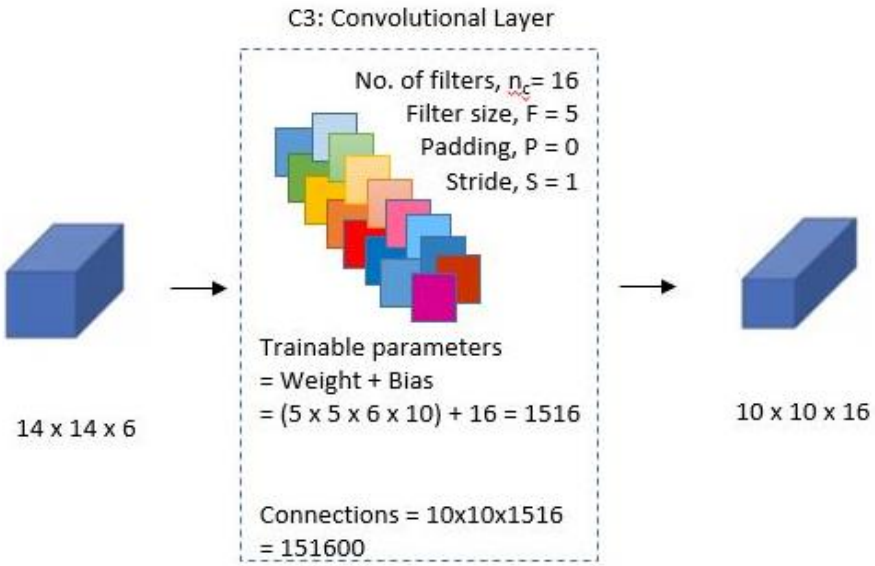


Рис. 2.5 Структура третього шару[14]

Четвертий шар. Четвертий шар (S4) знову є середнім шаром об'єднання з розміром фільтра 2 × 2 і кроком 2. Цей шар є таким же, як другий шар (S2), за винятком того, що він має 16 характеристичних карт, так що вихід буде зменшений до 5x5x16 (рис. 2.6).

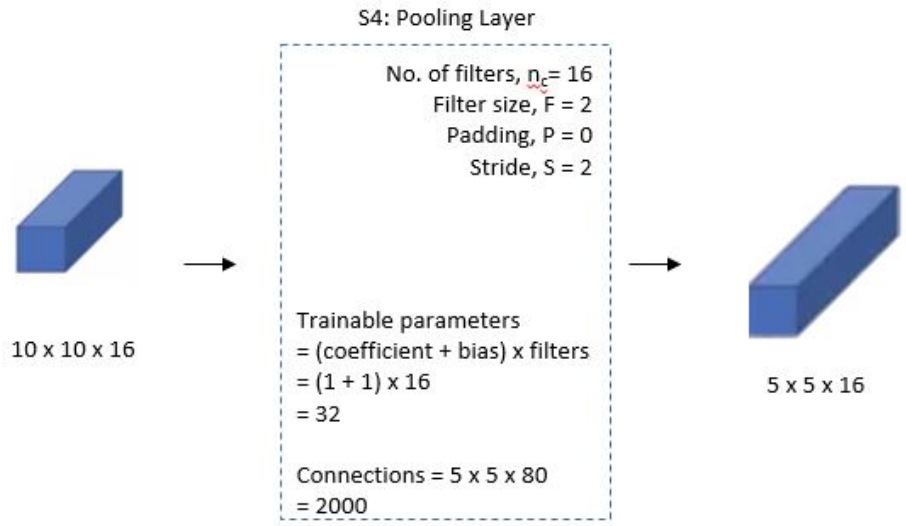


Рис. 2.6 Структура четвертого шару[14]

П’ятий шар. П'ятий шар (C5) є повністю пов'язаним згортковим шаром з 120 картами характеристик, кожен з розмірів 1 × 1. Кожен з 120 одиниць C5 з'єднаний з усіма 400 вузлами (5x5x16) в четвертому шарі S4 (рис. 2.7).

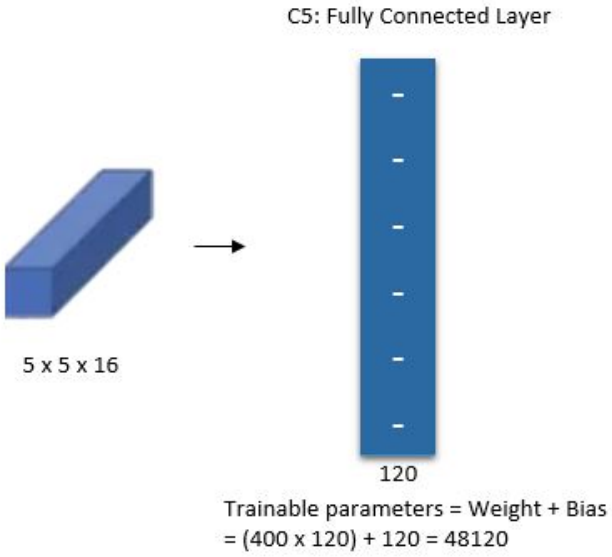


Рис. 2.7 Структура п'ятого шару[14]

Шостий шар. Шостий шар - це повністю з'єднаний шар (F6) з 84 одиницями (рис. 2.8).

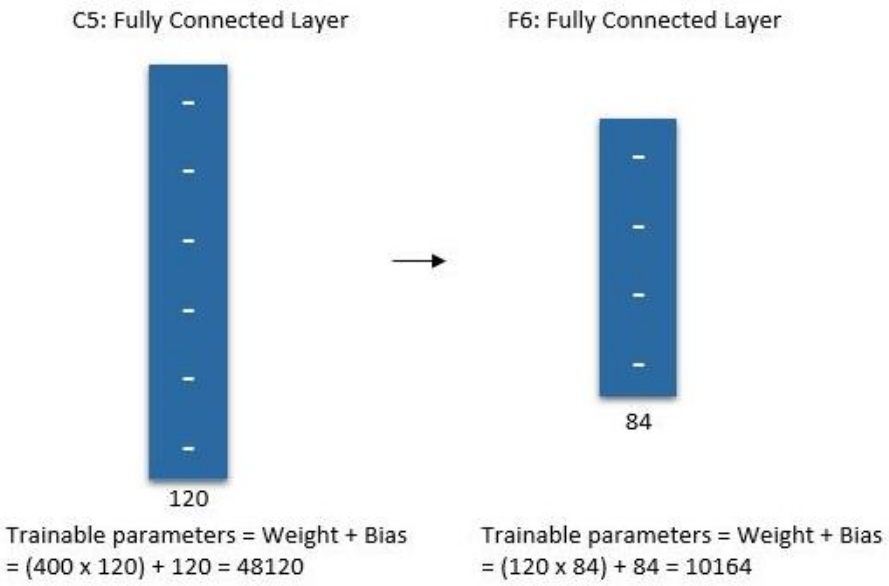


Рис. 2.8 Структура шостого шару[14]

Нарешті, є повністю пов'язаний вихідний шар softmax ŷ з 10 можливими значеннями, що відповідають цифрам від 0 до 9 (рис. 2.9).

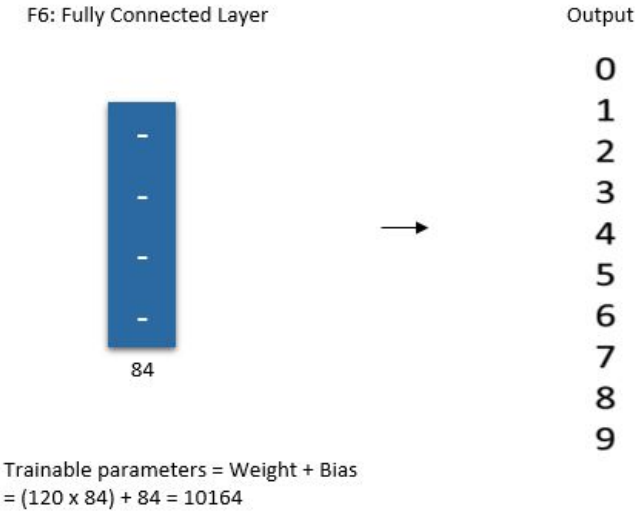


Рис. 2.9 Структура вихідного шару[14]

2.4.2 AlexNet

AlexNet – згорткова нейронна мережа, яка справила великий вплив на розвиток машинного навчання, особливо на алгоритм комп’ютерного зору.[4] Мережа виграла конкурс з розпізнавання зображень у 2012 році з кількістю помилок 15.3%. Схожа за архітектурою на мережу Яна ЛеКуна LeNet. Однак, у AlexNet більше фільтрів на шарі та вкладених згорткових

шарів. Мережа включає у себе згортки, максимальне об’єднання, дропаут, аугментацію даних, функцію активації ReLU та стохастичний градієнтний спуск.

Особливості AlexNet:

1) Функція активації використовується ReLU замість арктангенса для додавання у модель нелінійності. За рахунок цбого при однаковій точності методу швидкість стає більшою у 6 разів.
2) Використання дропауту замість регуляризації вирішує проблему перенавчання. Однак, час навчання подвоюється з показником дропауту 0.5

Архітектура мережі приведена на рис. 2.10. Вона складається з восьми шарів з ваговими коефіцієнтами. Перші п’ять з них згорткові, а інші три – повнозв’язні. Вихідні дані пропускаються через функцію втрат softmax, яка формує розподіл 1000 міток класів. Мережа максимізує багатолінійну логістичну регресію, що еквівалентно максимізації середнього по всім навчальним випадкам логарифма ймовірності правильного маркування з розподілу очікування. Ядра другого, четвертого і п'ятого згортальних шарів пов'язані тільки з тими картами ядра в попередньому шарі, які знаходяться на одному і тому ж графічному процесорі. Ядра третього згортвого шару пов'язані з усіма картами ядер другого шару. Нейрони в повнозв'язних шарах пов'язані з усіма нейронами попереднього шару.

Таким чином, AlexNet містить 5 згортальних шарів і 3 повнозв'язних шарів. Relu застосовується після кожного згортковго і повнозв'язного шару. Дропаути застосовуються перед першим і другим повнозв'язними шарами. Мережа містить 62,3 мільйона параметрів і витрачаєте 1,1 мільярда обчислень при прямому проході. Згорткові шари, на які припадає 6% всіх параметрів, виробляють 95% обчислень.

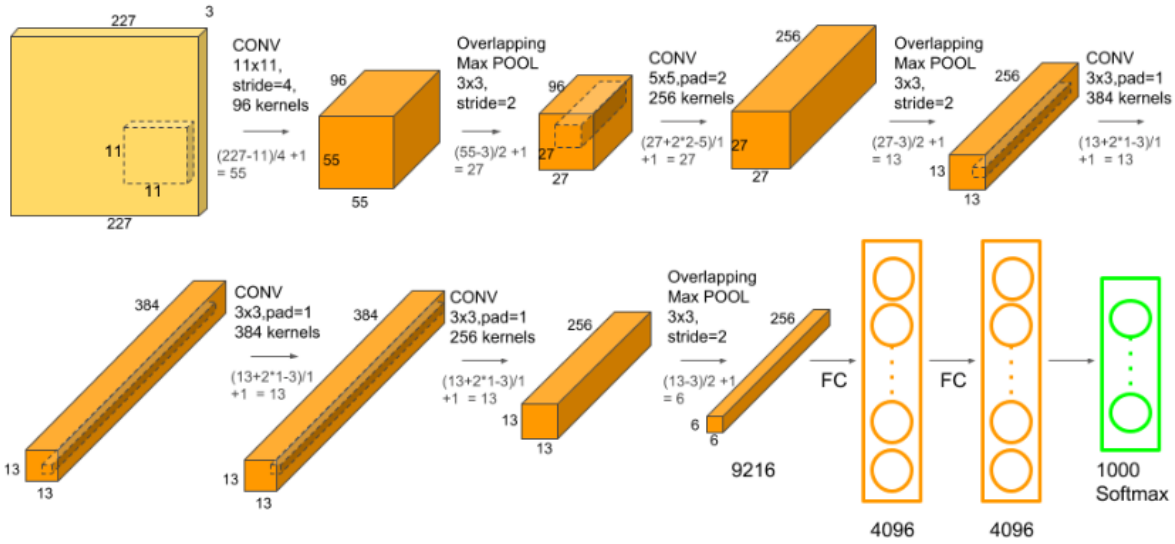


Рис. 2.10 Архітектура мережі AlexNet[4]

2.4.3. VGG-16

VGG-16 - модель згорткової нейромережі, яку запропонували вчені з Оксфордського університету. Модель досягає точності 92.7%. Ця нейромережа являється покращеною версією AlexNet, у якій замінені фільтри(розміру 11 та 5 у першому та другому згорткових шарах відповідно) на декілька фільтрів розміру 3 х 3, які йдуть один за одним[7].

Архітектура цієї нейромережі надана на рис. 2.11. На вході подаються RGB зображення розміру 224x224. Далі ці зображення проходять через стек згорткових шарів, у яких використовуються фільтри з дуже маленьким рецептивним полем розміру 3x3(який являється найменшим розміром для отримання представлення про те, де знаходиться право-ліво, верх-низ, центр).

У одній з конфігурацій використовується згортковий фільтр розміру 1x1, який може бути представлений, як лінійна трансформація вхідних каналів. Згортковиій шар фіксується на значенні 1 піксель. Просторове доповнення входу згорткового шару вибирається таким чином, щоб просторове розширення зберігалося після згортки, тобто доповнення

дорівнює 1 для 3x3 згорткових шарів. Просторовий пулінг виконується з допомогою п’яти max-pooling шарів, які йдуть за одним із згорткових шарів. Операція max-pooling виконується на вікні розміром 2x2 пікселя з кроком 2.

Після стеку згорткових шарів йдуть три повнозв’язних шари: перші два мають по 4096 каналів, а третій – 1000. Останнім йде soft-max шар. Конфігурація повнозв’язних шарів одна й та ж у всіх нейромережах.

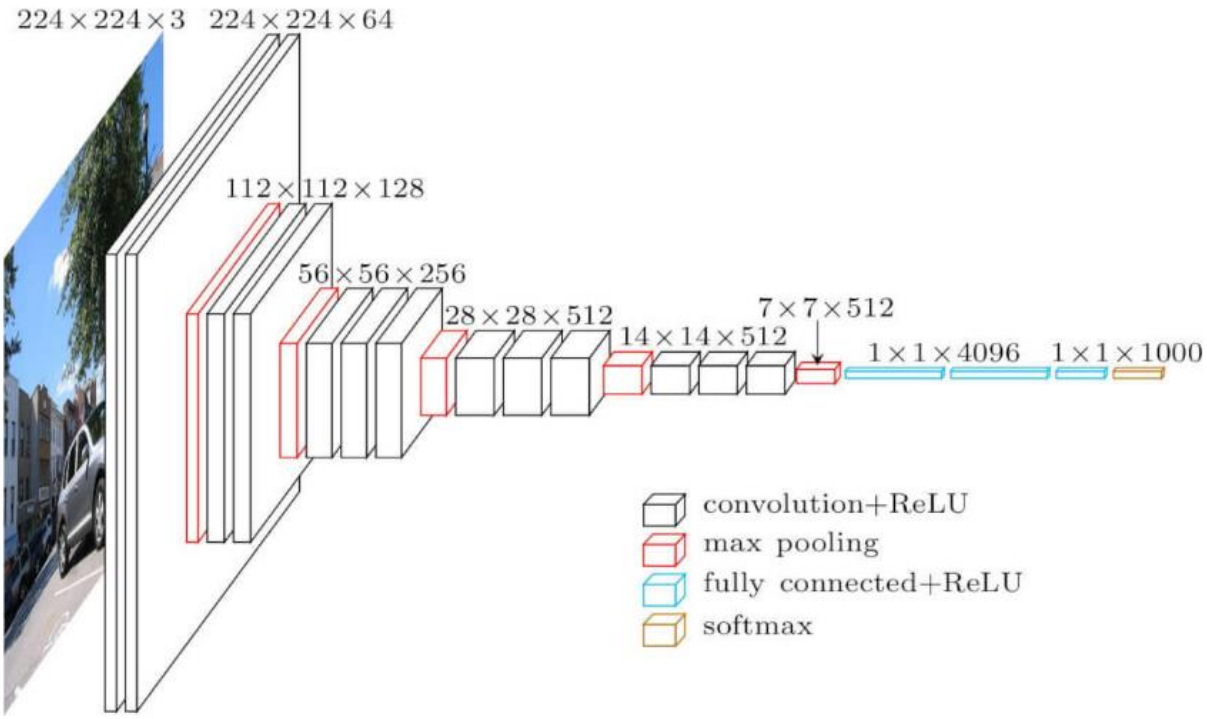


Рис. 2.11 Архітектура нейромережі VGG-16[7]

2.5 Алгоритм інструменту перевірки унікальності графічного зображення

Алгоритм динамічного завантаження було організовано як позначено на рис 2.12.

Рис. 2.12 Схема алгоритму роботи інструменту перевірки унікальності графічного зображення

Нове зображення надходить до системи. Після цього з допомогою фреймворку зображення нормалізується та переводиться до виляду, який зручний для обробки нейромережею. Нейромережа пропускає через шари зображення та на виході віддає вектор характеристик завантажуваного зображення.

Для того, щоб дізнатися наскільки вхідне зображення унікальне, потрібно перевірити схожість з усіма уже завантаженими раніше зображеннями. Для збереження часу та ресурсів використовується база даних, до якої зберігаються вектори характеристик зображень. Замість того, щоб проганяти кожного разу через нейромережу усі зображення, вектори завантажуються з бази.

Щоб зробити висновок про унікальність зображення, потрібно знайти серед усіх знайжених консинусоїдальних відстаней найбільшу та порівняти з пороговим значенням. Воно формується на базі того, наскільки точною повинна бути копія зображення, наскільки допустиме його трансформація та інше.

Після перевірки максимальної відстані якщо зображення унікальне, то воно завантажується до сховища та вектор його характеристик записується до бази даних. Якщо ж зображення не унікальне, то робота алгоритму завершується.

2.6 Аналіз ефективності методу перевікри унікальності графічних зображень

Під час перевірки унікальності графічних зображень дуже часто постає проблема точності обробки зображень та часу виконання операцій. Це відбувається здебільшого через затрачений час на нормалізацію зображення та приведення його до вигляду, який буде сприйнятливий для оцінки.

Точність обчислення залежить від алгоритму виявлення характеристик оцінюваного зображення та порівняння цих характеристик з існуючими для

інших зображень. Для того, щоб точніше виявити схожість, потрібно використовувати багато обчислень, щоб перевірити всі можливі перетворення зображення.

Перевірка унікальності графічних зображень з допомогою нейромережі дозволяє значно скоротити час через збереження вектору характеристик, з якими працює нейромережа, до бази даних. Також точність здобувається тим, що архітектура згорткової нейромережі направлена на визначення особливостей кожного зображення та збереження їх до вектора характеристик, що допомагає при знаходженні косинусоїдальной відстані між векторами.

ВИСНОВКИ ДО РОЗДІЛУ 2

В цьому розділі буди оглянуті актуальні мови програмування та середовища розробки задля вибору найбільш зручної для вирішення обраної проблеми. Ними стали мова Java та середовище розробки intellij IDEA.

Було детально розглянуто різні архітектури згорткових нейромереж, їх переваги та недоліки, а також фреймворки, з допомогою яких буде вестися робота з нейромережею. Фреймворком був обраний Deeplearning4j та архітектурою була обрана VGG-16. Був розроблений алгоритм перевірки унікальності графічних зображень з допомогою нейромереж та збереженням попередніх характеристик до бази даних, а також була проаналізована його ефективність.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ВИРІШЕННЯ ЗАДАЧІ ТА РОЗРОБКА ПРОГРАМИ

3.1 Модель перевірки унікальності графічних зображень

Перевірка унікальності графічних зображень повинна базуватися на перетворенні зображення до такого, з яким можна швидко та зручно працювати та на визначенні специфічних характеристик вхідного зображення і порівнянні їх з вже уснуючими у сховищі.

У типових алгоритмах пошуку нечітких дублікатів зображення для врахування різних перетворень зображення потрібно враховувати різні метрики або враховувати декілька гістограм. Метод, що базується на нейромережах простіший та точніший, на відміну від стандартних. Використання фреймворку ще більш спрощує та пришвидшує вирішення поставленої задачі.

Під час перевірки на унікальність зображення з допомогою нейромережі, достатньо один раз визначити вектор характеристик вхідного зображення та зберегти у базу даних для подальшої роботи з ними. Таке рішення пришвидшує обробку даних вже існуючих у сховищі зображень. Архітектура нейромережі VGG-16 за рахунок багатошаровості, специфічності розташування шарів та методу зворотнього поширення помилки може дуже точно визначати специфіку кожного зображення, що відображається на більш точних визначеннях різниці між характеристиками вхідного та уже існуючих зображень.

3.2 Реалізація методу перевірки унікальності графічних зображень

Отже, метод має такі етапи:

1) Завантаження даних нейромережі: завантаження моделі нейромережі та її даних

2) Обробка вхідного зображення: завантаження нового зображення, нормалізація зображення, обробка зображення нейромережею, отримання вектору характеристик
3) Аналіз схожості: знаходження максимальної відстані між векторами характеристик
4) Рішення про завантаження: на основі аналізу вирішується чи буде завантажене нове зображення до сховища

3.2.1 Реалізація завантаження даних нейромережі

Для роботи з нейромережею було обрано фреймворк Deeplearning4j. На початку завантаження системи проходить ініціалізація всіх компонентів та настроювання середовища для роботи з моделлю нейромережі. Так як було обрано архітектуру VGG-16, на початку потрібно створити екземпляр класу даної архітектури з допомогою конструктора new VGG16().

Після створення нового екземпляра моделі, потрібно завантажити зазадлегідь протреновану мережу. Фреймворк надає можливість завантажити екземпляр мережі та його дані, що були треновані на базі великої колекції зображень ImageNet. Після завантаження всіх даних додаток остаточно завантажується та можна переходити до основного алгоритму перевірки унікальності графічних зображень та завантаження їх до локального сховища.

3.2.2 Реалізація обробки вхідного зображення

Перш за все потрібно завантажити нове зображення через веб-інтерфейс. В даному випадку, вхідне зображення прийде до методу handleFileUpload() контроллера FileUploadController.java (рис. 3.1).

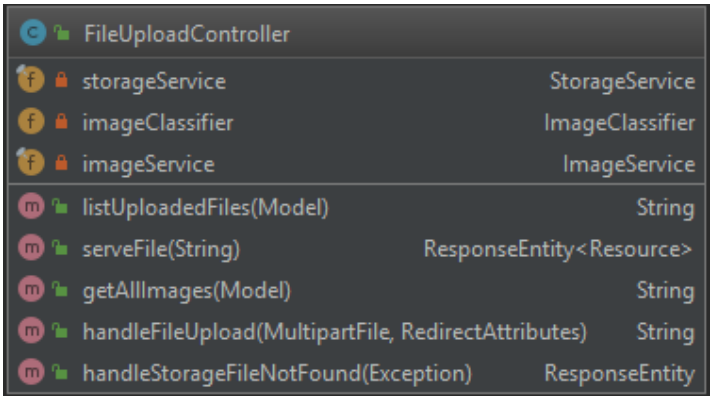


Рис. 3.1 Реалізований клас FileUploadController.java

Після завантаження фото, контроллер передає повноваження роботи з вхідним зображенням класу ImageClassifier.java (рис. 3.2).

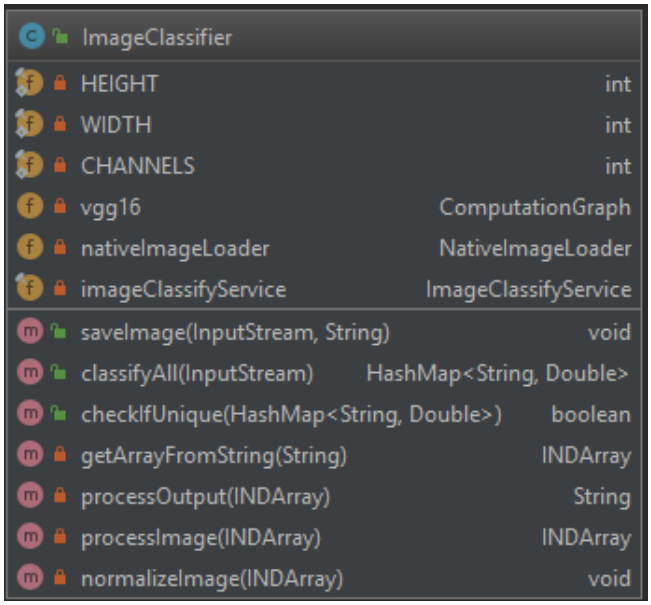


Рис. 3.2 Реалізований клас ImageClassifier.java

У класі ImageClassifier.java приймає вхідне зображення метод classifyAll(). Метод слугує для основних операцій обробки вхідного зображення. Спочатку з допомогою методу normalizeImage() зображення приймає сприйнятливий для подальшої обробки нейромережею вигляд. Після нормалізації викликається метод processImage(), який відправляє зображення безпосередньо до нейромережі для виділення вектору характеристик.

Після отримання вектору характеристик вхідного зображення, викликається метод findAll() сервісу ImageService.java(рис. 3.3) для отримання усіх векторів характеристик існуючих зображень з бази даних.

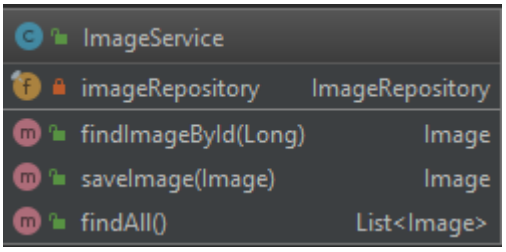


Рис. 3.3 Реалізований клас ImageService.java

Після цього у циклі знаходиться косинусоїдальна відстань між вектором характеристик вхідного зображення та вожним вектором характеристик зображень зі сховища, що були взяті із бази даних. На виході отримуємо HashMap<String, Double>, яка містить назву зображення та міру подібності до нього вхідного зображення.

3.2.3 Реалізація аналізу схожості

Після попередніх операцій отримується набір мір подібності вхідного зображення до тих, що були уже завантажені до сховища. Для того, щоб перейти до аналізу, потрібно відшукати максимальну серед відстаней, що представлені у наборі. Цим буде займатися метод checkIfUnique() класу ImageClassifier.java(див. рис. 3.2).

Отримавши максимальну з відстаней, можна робити висновок про унікальність зображення. Існує статичне значення порогу схожості. Воно визначається імпірично, залежно від бажаної чутливості до модифікацій зображення. Чим більша відстань, тим більш схожі зображення між собою. Відстань між векторами лежить у діапазоні від 0 до 1. Отже, чим ближче значення до 0, тим більш унікальне зображення.

У методі перевіряється умова , за якаою порівнюється максимальна відстань між векторами з порогом схожості. Якщо більше порогу, то зображення уже існує у сховищі, якщо ж ні – унікальне.

3.2.4 Реалізація рішення про завантаження

З допомогою попереднього аналізу було виявлено чи унікальне зображення, чи ні. Якщо ж зображення унікальне, то з допомгою методу saveImage() сервісу ImageService.java(див. рис. 3.3) буде отримано вектор характеристик вхідного зображення та збережено його до бази даних. Після збереження до бази з допомгою методу store() сервісу FileSystemStorageService.java(рис. 3.4) буде завантажено зображення до локального сховища.

Якщо ж зображення було визнано не унікальним, то контроллер поверне на веб-сторінку повідомлення про невдалу спробу завантаження та про те, що таке зображення уже існує у сховищі.

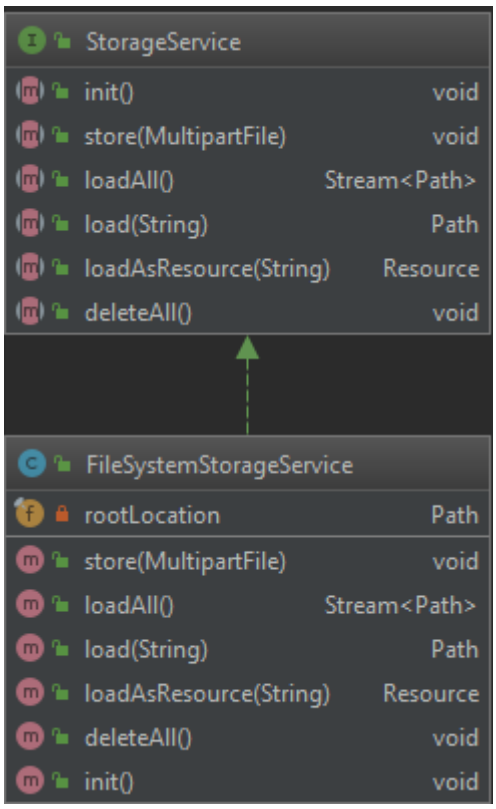


Рис. 3.4 Реалізований інтерфейс StorageService.java та клас FileSystemStorageService.java

3.3 Інструкція користувачеві

Розроблювана система має вигляд веб-додатку, що працює локально на порту 8080, тому для того, щоб відкрити сторінку потрібно зайти у будь-який браузер та у адресному рядку ввести localhost:8080.

На даному етапі розробки додаток має єдиний екран – головну сторінку(рис. 3.5), на якій знаходиться кнопка «Вибрати файл» для перегляду файлової системи та завантаження зображення. Під нею знаходиться кнопка «Завантажити».

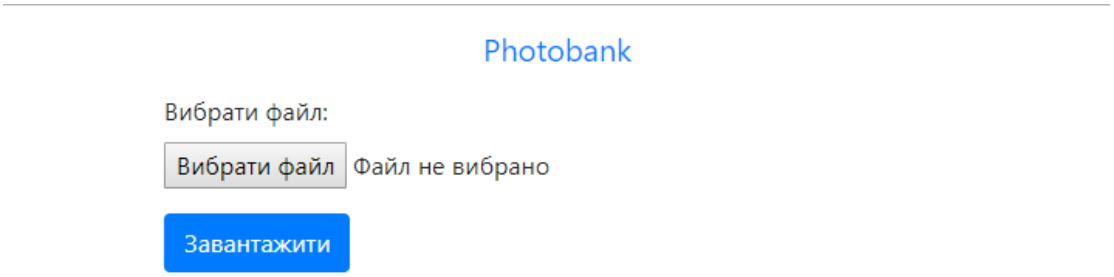


Рис. 3.5 Початковий екран системи

Під час натискання на кнопку «Вибрати файл» буде відкрито «Провідник» та користувач може вибрати будь-який файл з розширенням «.jpeg» або «.png» розміром до 10 MB.

Після обрання зображення для завантаження до сховища потрібно натиснути на кнопку «Завантажити». Після цього відбудеться основна робота алгоритму та буде вирішено чи унікальне зображення. Якщо зображення буде визнане унікальним, то з’явиться у текстовому полі повідомлення з текстом «Зображення успішно завантажено до сховища!»(рис. 3.6), якщо ж ні, то з текстом «Зображення не унікальне. Спробуйте ще!»(рис. 3.7). Під текстовим полем буде відображатись зображення, яке було обрано та перевірено на унікальність.

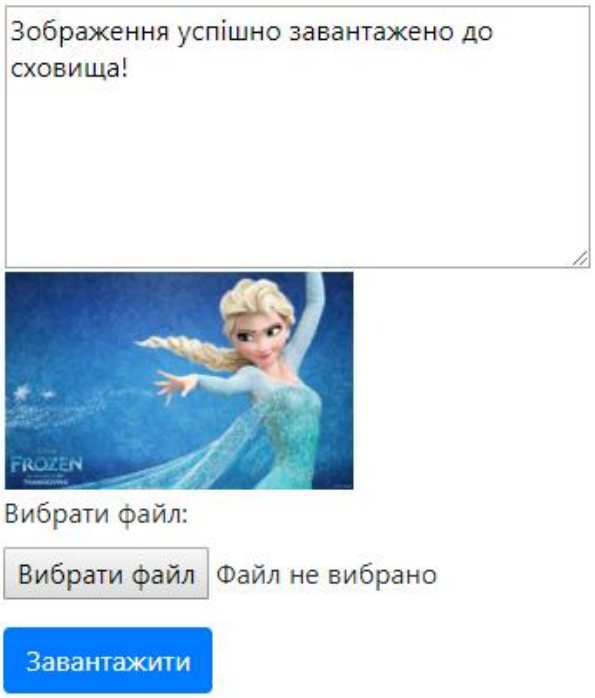


Рис. 3.6 Успішне завантаження зображення

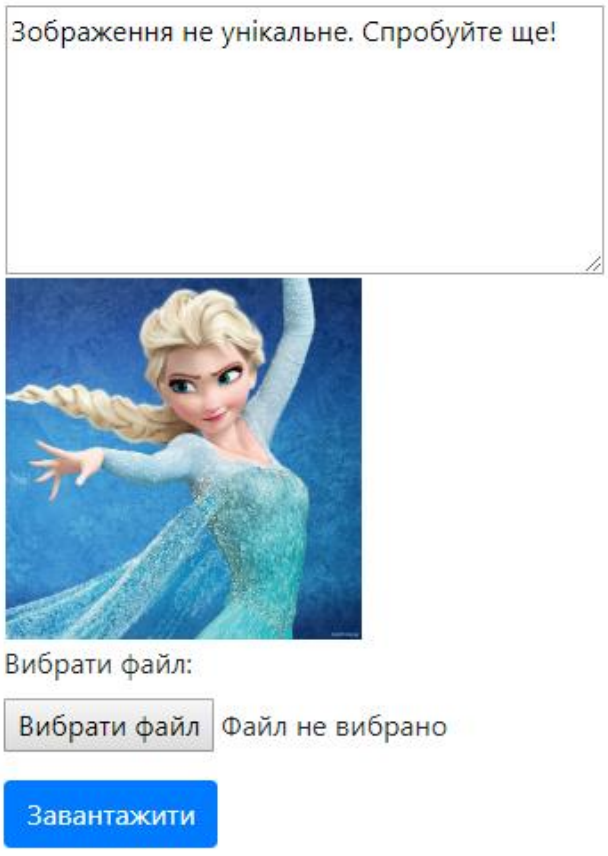


Рис. 3.7 Зображення не унікальне

Під зображенням, незалежно від результату завантаження, буде зберігатися набір кнопок для завантаження нового зображення, тому користувач без переходу на інший екран може продовжувати роботу з завантаженням зображень до сховища.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було описано реалізацію алгоритму перевірки унікальності графічного зображення на унікальність. Було розглянуто етапи роботи алгоритму та детально описано кожен з них. Було описано інтеграцію роботи з нейромережею у існуючому алгоритмі та взаємодію з базою даних при збереженні та отриманні даних для обробки. Після опису алгоритму надано інструкцію користувача, яка демонструє працездатність запропонованого алгоритму.

ВИСНОВКИ

Дана робота була розроблена з метою вирішення задачі оптимізації завантаження графічних зображень до сховища та перевірки їх унікальності. Було запропоновано використання технології нейромереж для вирішення поставленої задачі.

У відповідних розділах було розглянуто згорткові нейромережі, їх структуру та виконано аналіз кожного типу шарів нейромережі. Було оглянуто недоліки та переваги згорткових нейромереж. Також було проаналізовано можливі рішення проблеми перевірки унікальності графічних зображень.

Під час проектування системи оглянуто декілька мов програмування, середовища розробки програмного забезпечення та допоміжні технології для більш комфортної взаємодії з базою даних і компонентами системи. Було обрано фреймворк, на основі якого базується робота з моделлю нейромережі. Розглянуті різні архітектури згорткових нейромереж та була обрана VGG-16 через найточніші показники.

Розроблено алгоритм методу перевірки унікальності графічних зображень та розглянуті його особливості і проаналізовано його ефективність. Описано реалізацію розробленого алгоритму. Описані головні класи системи та їх методи. Також було детально описано підготовку до роботи з моделлю нейромережі та кожен етап алгоритму методу перевірки унікальності графічних зображень.

В роботі запропоновано інструкцію користувача. Розроблений програмний комплекс реалізує наведений алгоритм перевірки унікальності графічних зображень та завантаження їх до локального сховища.

ЛІТЕРАТУРА

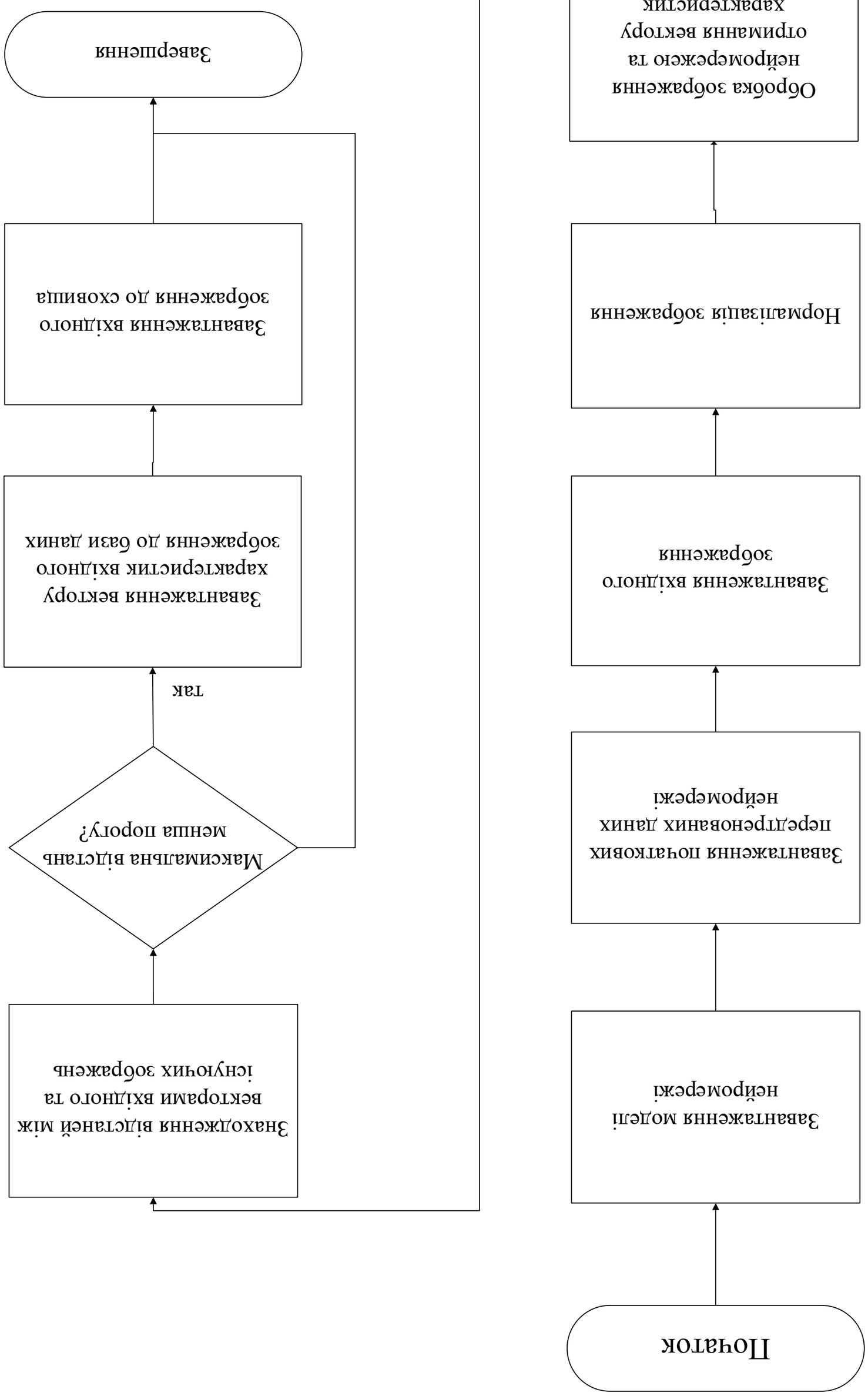
1. Yann LeCun. LeNet-5, convolutional neural networks [Електронний ресурс]. Режим доступу: http://yann.lecun.com/exdb/lenet/ (дата звернення: 23.03.2019)
2. Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход. Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 928с.
3. Antonio Torralba, Rob Fergus and William T. Freeman 80 million tiny images: a large dataset for non-parametric object and scene recognition [Електронний ресурс]. Режим доступу: http://people.csail.mit.edu/torralba/publications/80millionImages.pdf (дата звернення: 15.04.2019)
4. AlexNet — свёрточная нейронная сеть для классификации изображений [Електронний роесурс]. Режим доступу: https://neurohive.io/ru/vidy-nejrosetej/alexnet-svjortochnaja-nejronnaja-set-dlja-raspoznavanija-izobrazhenij/ (дата звернення 15.04.2019)
5. Добров Г. Б., Пятков Е. А. Алгоритм поиска нечетких дубликатов на основе простых признаков [Електронний ресурс]. Режим доступу: http://romip.ru/romip2009/ 07_hbc.pdf (дата звернення 17.04.2019)
6. Слесарев А.В., Мучник И.Б., Михалев Д.К., Крайнов А.Г., Котляров Д.И., Беляев Д.В. Поиск похожих изображений и дубликатов [Електронний ресурс]. Режим доступу: http://romip.ru/romip2010/ 12_yandex_cbir.pdf (дата звернення: 22.04.2019)
7. VGG16 — сверточная сеть для выделения признаков изображений [Електронний ресурс]. Режим доступу: https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/ (дата звернення: 20.04.2019)
8. Building REST services with Spring [Електронний ресурс]. Режим доступу: https://spring.io/guides/tutorials/rest/ (дата звернення: 20.04.2019)

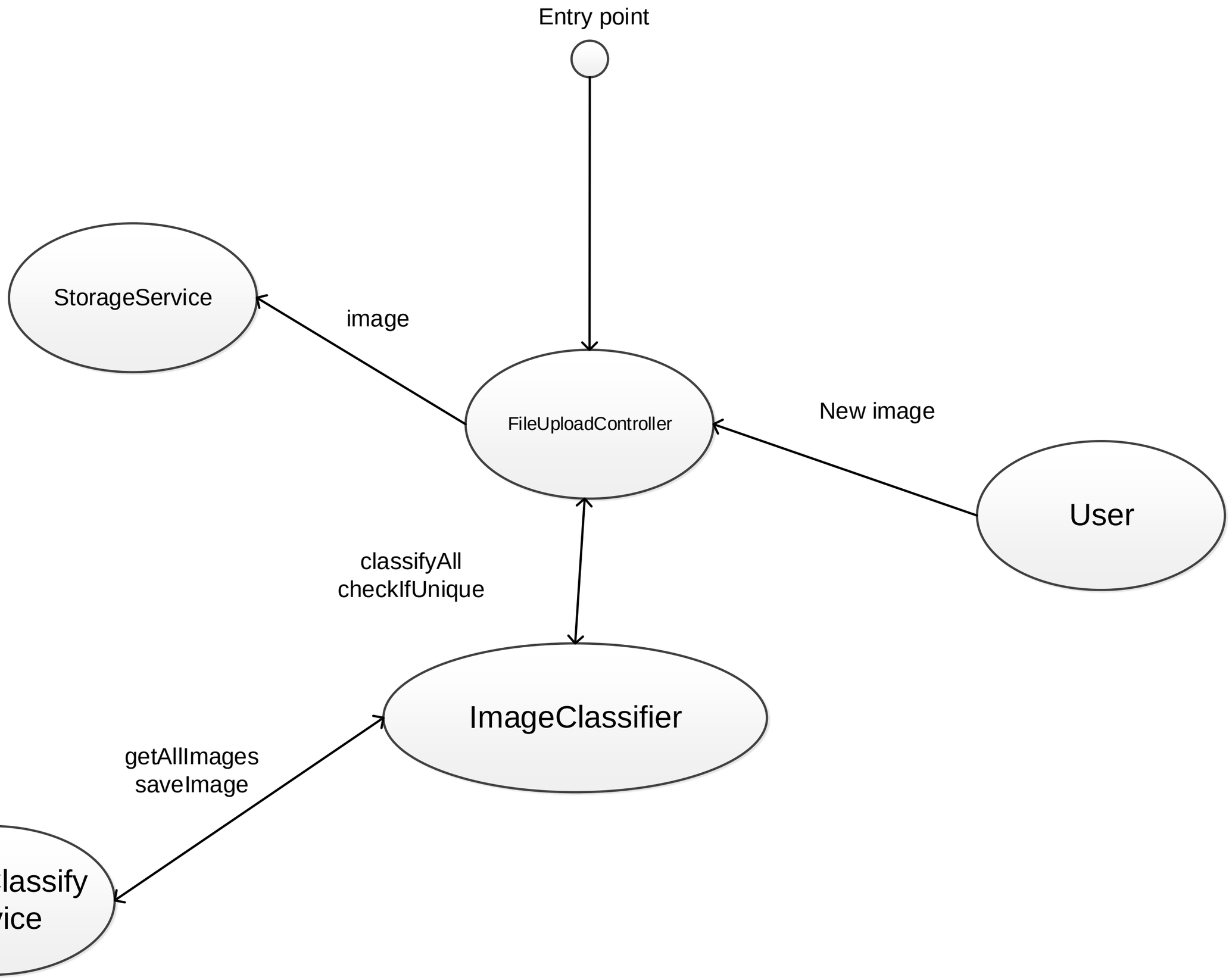
9. Neuroph. About Neuroph project [Електронний ресурс]. Режим доступу: http://neuroph.sourceforge.net/about_project.html (дата звернення: 20.04.2019)
10. Deeplearning4j. Guide [Електронний ресурс]. Режим доступу: https://deeplearning4j.org/docs/latest/ (дата звернення: 01.05.2019)
11. CNN Architectures [Електронний ресурс]. Режим доступу: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5 (дата звернення: 02.05.2019)
12. Згорткова нейронна мережа [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа (дата звернення: 01.04.2019)
13. Сравнение стеков Java EE и Spring: возможности и ограничения [Електронний ресурс]. Режим доступу: https://dou.ua/lenta/articles/javaee-vs-spring/ (дата звернення: 01.04.2019)
14. LeNet-5 – A Classic CNN Architecture [Електронний ресурс]. Режим доступу: https://engmrk.com/lenet-5-a-classic-cnn-architecture/ (дата звернення: 03.04.2019)

Графічні матеріали
до дипломного проекту

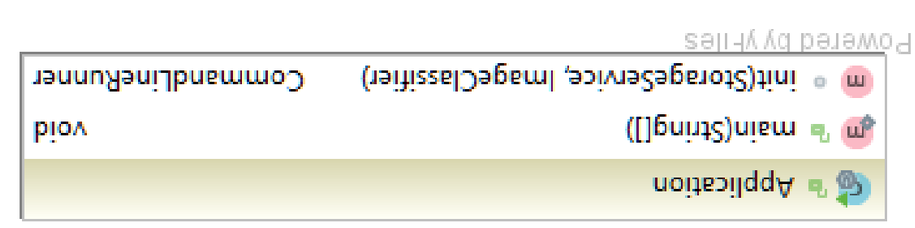
на тему: «Перевірка унікальності графічних зображень»

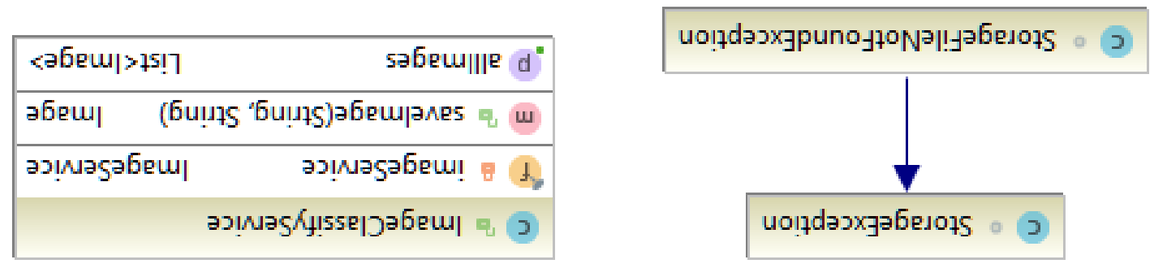
Київ – 2019

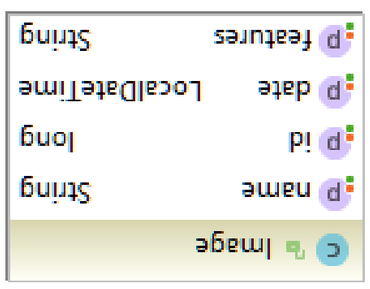

Початок
Завантаження моделі нейромережі
Завантаження початкових перетворених даних нейромережі
Нормалізація зображення
Завантаження вхідного зображення
Обробка зображення нейромережею та отримання вектору характеристик
Знаходження відстаней між векторами вхідного та існуючих зображень
Максимальна відстань менша порогу?
так
Завантаження вектору характеристик вхідного зображення до баз даних
Завантаження вхідного зображення до сховища
Завершення

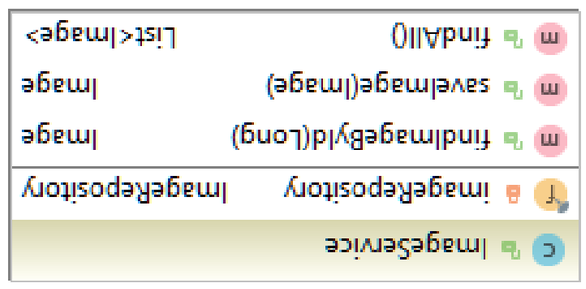


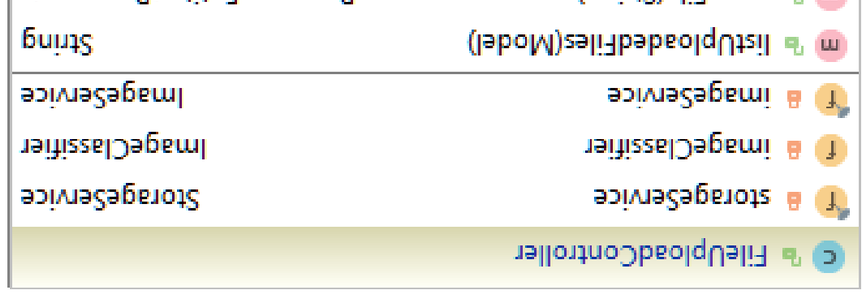
					ІАЛЦ.467200.005 Д2		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб .		Литвиненко А.С.			Перевірка унікальності графічних зображень. Схема взаємодії класів при перевірці на унікальність зображення	*Літ.*	*Арк.* 1 / *Аркушів* 1
Перевір.		Подрубайло О.О.					
						НТУУ „КПІ”, ФІОТ, ІП-53	
Н. Контр.		Сімоненко В.П.					
Затверд.		Стіренко С.Г.					

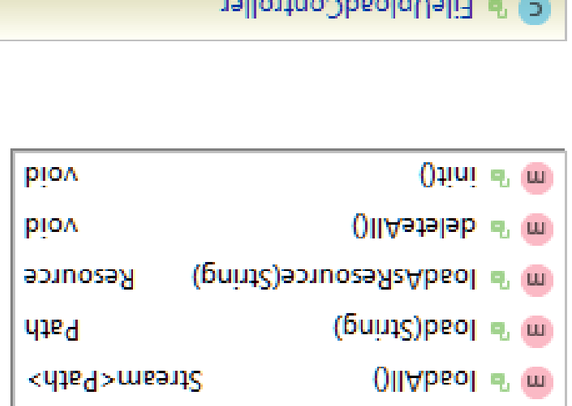


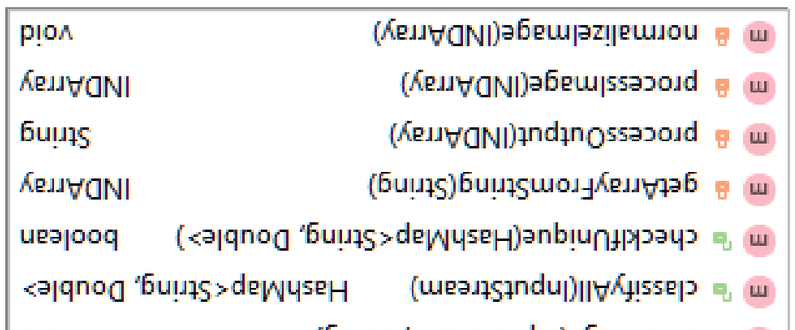












							ІАЛЦ.467200.006 Д3
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Литвиненко А.С.			Перевірка унікальності графічних зображень UML Діаграма класів програми	*Літ.*	*Аркушів*
Перевір.		Подрубайло О.О.					1
						1	
Н. Контр.		Симоненко В.П.				НТУУ „КПІ", ФІОТ, ІП-53	
Затверд.		Стрєнко С.Г.					

ДОДАТОК А

Перевірка унікальності графічних зображень

Текст програми

ІАЛЦ.467200.007 А1

Листів 11

Київ – 2019