Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

Магістерська робота

освітній ступінь - магістр

на тему: **«КЛАСИФІКАЦІЯ КОНФІДЕНЦІЙНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ»**

Виконав: студент 2-го року навчання,
Спеціальності
121 Інженерія програмного забезпечення
Нгуєн Сан Бинь Ванович

Керівник **Бучко О.А.**
кандидат технічних наук, доцент

Рецензент _____

Магістерська робота захищена з оцінкою
_____

Секретар ЕК _____
«___» _____ 2022 р.

Київ – 2022

Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

ЗАТВЕРДЖУЮ
Керівник магістерської роботи
_____ доцент Бучко О.А.
(підпис)
«____»_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на магістерську роботу

студенту 2-го року навчання, освітній ступінь магістр, факультету інформатики
Нгуєну Сану Биню Вановичу

ТЕМА: Класифікація конфіденційних зображень з використанням нейронних мереж

ЗАДАЧА: Створити систему аналізу та класифікації зображень для визначення рівня конфіденційності вмісту зображення (пошуку таких даних як паспортні дані, кредитні картки, документи тощо) використовуючи нейроні мережі.

ЗМІСТ ТЧ до магістерської роботи:

Зміст
Анотація
Вступ
1. Теоретичні аспекти алгоритму
2. Практичні аспекти алгоритму
3. Реалізація алгоритму
Висновки
Список літератури
Додатки

Дата видачі «___» _____ 2022 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання роботи

Тема: «Класифікація конфіденційних зображень з використанням нейронних мереж»

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	06.10.2021	
2.	Пошук та огляд літератури за темою роботи.	22.12.2021	
3.	Визначення структури практичної роботи та написання мінімального прототипу.	23.01.2022	
4.	Пошук та дослідження методів покращення алгоритму, вивчення альтернативних алгоритмів.	24.02.2022	
5.	Тестування алгоритму, оцінка алгоритму.	01.03.2022	
6.	Фінальне оформлення практичної частини.	07.04.2022	
7.	Написання пояснювальної роботи.	01.06.2022	
8.	Створення слайдів для доповіді та написання доповіді.	28.06.2022	
9.	Захист роботи	07.07.2022	

Студент *Нгуєн Сан Бинь Ванович*

Керівник *Бучко Олена Андріївна*

“___”_____

Зміст

Перелік прийнятих скорочень

ANN	–	Approximate Nearest Neighbor
CNN	–	Convolutional Neural Networks
CSAM	–	Child Sexual Abuse Material
DH	–	Deep Hashing
DHN	–	Deep Hashing Network
DPSH	–	Deep Pairwise-Supervised Hashing
L2H	–	Learning To Hash
LFH	–	Latent Factor Hashing
LSH	–	Locality-Sensitive Hashing
TBDH	–	Triplet-based Deep Hashing

Анотація

Запропонована робота пропонує алгоритм вирішення задачі класифікації зображень, а саме класифікацію зображень на основі вмісту конфіденційної інформації, та досліджує різні теоретичні аспекти пов'язані з нею.

Головна ідея алгоритму полягає у пошуку схожих зображень, як такі що будуть слугувати взірцем для визначення класів, та спирається на швидкий пошук таких схожих зображень. Зображення у свою чергу не будуть використовуватися безпосередньо для визначення подібності, оскільки предметна область накладає на це обмеження. Алгоритм використовує хеш-коди, що дозволяє забезпечити приватність світлин користувачів. Також робота вводить такий термін як «псевдоадаптивність» мережі, тобто такої, що може обробляти нові, ще не бачені, класи, та додавати до вже існуючих класів нові випадки, які не були передбачені на етапі навчання.

Перший розділ роботи досліджує теоретичні аспекти, пов'язані з машинним навчанням, а саме з нейронними мережами, різновидами навчання. Досліджується задача пошуку найближчого сусіда та хешування. Також визначається предмет дослідження і власне термін конфіденційності зображень.

Другий розділ розглядає практичні аспекти запропонованого алгоритму, досліджуються проблеми задачі, представлення зображень за допомогою хеш-кодів, та різноманітні алгоритми глибокого хешування.

Третій розділ передбачає вже практичне дослідження, в якому пропонується алгоритм вирішення поставленої задачі, описується набір даних, на якому відбувається навчання, а також розбираються бібліотеки та середовища, які використовуються для реалізації алгоритму.

Вступ

Як ніколи раніше, людство генерує величезну кількість інформації, використовуючи свої персональні пристрої - смартфони, ноутбуки, планшети. Зображення, відео, аудіо, текст - завантажуються на безліч різноманітних платформ, таких як соціальні мережі, месенджери, веб-сервіси та інші застосунки, що у свою чергу несе велику небезпеку для людини та для її персональної інформації. Конфіденційність користувача довгий час експлуатується в мережі інтернет - використовуючи такі прості речі, як вік, вага, національність, релігія, вподобання тощо, зацікавлені сторони заманюють потенційного клієнта у пастку пропозицій і послуг. Користувачі однак часто не розуміють небезпеку деяких дій, які розкривають їх персональну інформацію на показ. Така проста і швидка операція як завантажити фотографію в інтернет може спричинити значущої шкоди, якщо необережно поводитися зі своєю конфіденційною інформацією. Конфіденційна інформація, яка може міститися в особистих зображеннях, інколи не розпізнається їх користувачами як «небезпечна для розповсюдження», а тому може легко бути поширена в мережі самим власником без роздумів. Варто розглянути і іншу сторону таких дій, оскільки не тільки самого користувача треба захищати від такої інформації, а й інших від неї. Нерідко такий контент може бути чутливим для інших, особливо для деяких категорій користувачів.

Тому починають з’являтися системи захисту персональної інформації, що перешкоджають користувачеві, якщо той випадково захоче поширити небезпечну» інформацію. До таких систем вимагають особливі вимоги до конфіденційності, що часто ускладнює їх розробку. Системи пошуку конфіденційної інформації можуть не тільки перешкоджати користувачу та захищати його персональну інформацію, а й вселяти відчуття безпеки і приватності, оскільки можуть фільтрувати контент не тільки в мережі, а власне на особистому пристрої. Користувач може власноруч виставити ті категорії контенту, який він вважає «чутливим» або «приватним», щоб той навіть не відображався в інтерфейсі застосунку.

Наукове значення роботи полягає у застосуванні сучасних підходів класифікації, за допомогою нейронних мереж та алгоритмів пошуку найближчих сусідів, до такої прикладної задачі, як розпізнання конфіденційності зображень, та перевірки їх спроможності ефективно та точно виконувати поставлену задачу. Запропонований алгоритм буде втілювати ідеї таких досліджень, як Deep Hashing, LSH, Siamese Network тощо. Також буде запропоновано алгоритм визначення приналежності запиту до конкретних класів, за допомогою визначення його найближчих сусідів, та загальний ступінь конфіденційності зображення на основі користувацьких налаштувань щодо обраних класів конфіденційності.

Мета цієї роботи розробити систему, яка буде класифікувати конфіденційний контент на зображеннях, щоб уникнути їх розповсюдженню і перегляду. Система мусить бути адаптивною, тобто підлаштовуватися під нові класи, які можуть додавати користувачі, що накладає на систему вимогу постійного навчання. Від системи буде вимагатися приватність, адже ніхто, крім власне користувача, не мусить мати доступу до перегляду особистих фотографій. Також вимагається швидкість обробки, оскільки в сучасних реаліях середня кількість фото у бібліотеках користувачів становить приблизно 5 - 10 тисяч світлин, в одну секунду робиться приблизно 38 фотографій на всі активні смартфони світу [1], а кількість зображень, які зберігаються на серверах популярних соціальних мереж може сягати 240 мільярдів [2].

Розділ 1. ТЕОРЕТИЧНІ АСПЕКТИ АЛГОРИТМУ

1.1 Конфіденційна інформація

Конфіденційна інформація – (від англ. confidence — довіра) це відомості, які знаходяться у володінні, користуванні або розпорядженні окремих фізичних чи юридичних осіб і поширюються за їх бажанням.

Згідно з різними законодавчими документами України (Закон України, Податковий кодекс України тощо) до конфіденційної інформації фізичної особи відносять наступне: дані про національність, релігію, освіту, сімейний стан, стан здоров'я, дата і місце народження, місце проживання, місце реєстрації, дані про особисте життя громадян, прибутки, документи що посвідчують особу, особисті документи тощо.

Пануючі тенденції у розробці програмних застосунків все частіше піднімають тему конфіденційності користувачів та їх даних. Прикладом є компанія Apple, яка останні два роки сконцентрувалася на інформаційній безпеці і підвищені конфіденційності користувачів власних продуктів. Основна ціль застосунків, це збільшення надійності систем, а отже і інформації, яка належить користувачам, і безпека щодо нерозповсюдженню їх інформації.

1.1.1 Конфіденційність зображень

Фотографії (або персональні зображення користувача) часто слугують джерелом конфіденційної інформації. Це може бути фото паспорту, де вказано повне ім'я, дата та місце народження, органи які видали документ, підпис особи і тд. Таку інформацію шукають зловмисники, щоб здійснити злочини, або просто проводити соціальний інжиніринг. Часто, людина не замислюється, що фото зроблене нею містить дуже важливу і конфіденційну інформацію про неї (див. Рисунок 1.1), і може випадково поширити цю фотографію у своїх соціальних мережах, месенджерах тощо. Свідомі користувачі часто складають свої конфіденційні фотографію в особливу папку на їх девайсах, яка ховає такі фотографії і зайвий раз убезпечує їх від

випадкового поширення, оскільки часто застосунки не мають доступ до таких фотографій. Світлини можуть мати і чутливий або делікатний характер, що теж небезпечно поширювати.

Рисунок 1.1 − Приклад зображення з конфіденційним вмістом (номер кредитної картки) [12].

1.2 Пошук найближчого сусіда

Пошуку найближчого сусіда (англ. Nearest Neighbor Search (NNS)) — це пошук елемента, який є найближчим або найбільш схожим до заданого елемента, у деякому наборі даних. У задачі пошуку найближчого сусіда задано деякий запит (точка) $q \in R^d$ та набір даних з N елементів (точок) $X = \{x_1, x_2, \ldots, x_N\} \subset R^d$. Ціль − знайти найближчу точку $x = argmin_x distance(x, q)$ до запиту q відповідно до деякої міри відстані.

Для маловимірних просторів задача пошуку найближчого сусіда вже має визначені алгоритми, наприклад, методи індексування за допомогою дерев, такі як KD-дерево, які засновані на рекурсивно-структурованому розбитті простору даних.

Більшість сучасних програм мають масивні набори даних з високою розмірністю (сотні чи тисячі). Для багатовимірних даних ці алгоритми стають непрактичними, оскільки потребують складної та великої структури зберігання, та мають досить низьку швидкість. Такі обмеження як розумне споживання пам’яті та/або низька затримка, змушують поступитися точністю результатів.

Важливо зазначити, що незважаючи на всі останні досягнення в цій темі, єдиним методом пошуку точного найближчого сусіда є вичерпний пошук.

Тому існують спрощені варіанти цієї задачі, які називаються наближеним пошуком найближчого сусіда (англ. Approximate Nearest Neighbor (ANN)).

1.2.1 Наближений пошук найближчого сусіда

Існує декілька різновидів наближеного пошуку найближчого сусіда, однак основна ідея зберігається у всіх така – знайти деяку кількість найближчих сусідів з заданою похибкою.

У $(1+\varepsilon)$ наближеному пошуку найближчого сусіда, при деякому запиті $q \in R^d$ та наборі даних $X = \{x_1, x_2, \ldots, x_N\} \subset R^d$, достатньо знайти точку даних $x^* \in X$ таку, що $distance(x^*, q) \leq (1+\varepsilon) distance(x', q)$, де x' – справжній найближчий сусід відносно заданої функції відстані $distance$, а $\varepsilon > 0$.

Задача пошуку R-ближнього сусіда має на меті знайти деякий елемент x, який називається R-ближнім сусідом, так що $distance(x, q) \leq R$.

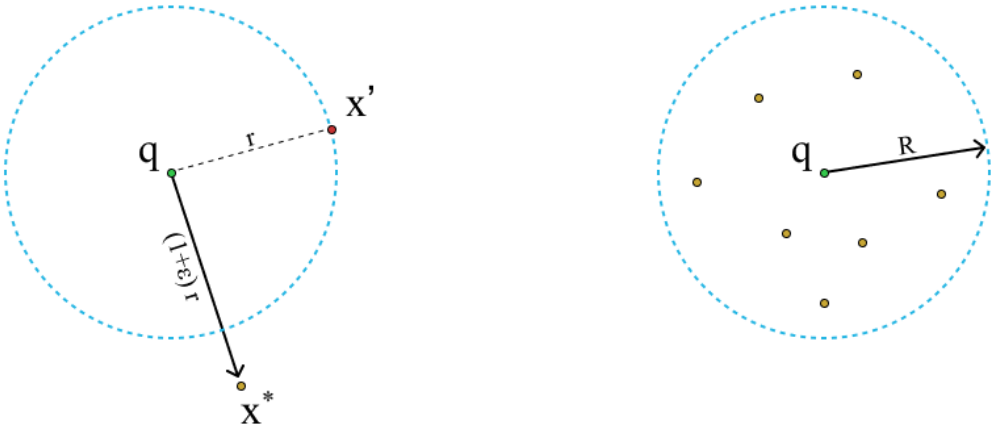


Рисунок 1.2 − Зліва ілюстрація задачі пошуку $(1+\varepsilon)$ наближеного найближчого сусіда, справа − задача пошуку R-ближнього сусіда.

Задача наближеного пошуку найближчого сусіда відіграє важливу роль в інформаційно-пошукових системах, оскільки для практичного застосування часто

достатньо отримати приблизних найближчих сусідів для заданого запиту, тобто елементи у базі даних, які достатньо подібні.

В останні роки пошук наближеного найближчого сусіда став помітною темою досліджень для ефективної обробки постійно зростаючого обсягу даних у практичних застосунках. ANN має безліч застосувань, включаючи розпізнавання шаблонів, системи рекомендацій, пошук схожості, кластерний аналіз тощо. Однак у цій роботі зосередимося насамперед на застосуванні пошуку схожих зображень. Крім того, серед існуючих методів ANN, хешування стало ефективно популярним методом для зберігання та обробки даних великої розмірності. Сталося це завдяки високій швидкості запитів і низьким витратам пам'яті.

Методи наближеного пошуку найближчого сусіда прискорюють пошук шляхом попередньої обробки даних у ефективний індекс, і часто вирішуються за допомогою таких етапів:

Векторне перетворення — застосовується до векторів перед їх індексацією, серед них є, наприклад, зменшення розмірності.

Векторне кодування — застосовується до векторів для побудови фактичного індексу для пошуку, серед них є методи на основі структури даних, такі як дерева, LSH і квантування — техніка для кодування вектору в набагато більш компактній формі.

Варіанти використання «найближчого сусіда» нескінченні, задачу можна зустріти в багатьох областях комп'ютерних наук, таких як розпізнавання зображень, машинне навчання та обчислювальна лінгвістика. Задачу пошуку найближчого сусіда можна використовувати для розпізнавання образів, в системах рекомендацій, та для класифікації зображень або текстів.

У запропонованій роботі задача пошуку найближчого сусіда буде використовуватись для класифікації зображень, де «сусід» буде виступати як зразкове зображення для визначення класу конкретного зображення.

1.3 Хешування

Хеш-функція — це деяка функція, яка відображає елементи довільного розміру в деяке фіксоване значення в інтервалі [0, m]. Умовно позначимо хеш-функцію як $h(x) = y, x \in R^d, y \in Z$, де h - хеш-функція, x - деякий елемент з набору даних R^d, та y - хеш-значення, або просто хеш. Зазвичай хеш-функція використовується для індексації даних в так звані хеш-таблиці. Хеш може задаватися в довільній формі - це може бути ціле число, представлення деякого числа в двійковій, восьмирічній, шістнадцятирічній системах, як масив деяких чисел тощо, але всі ці форми можна звести до представлення деякого цілого числа.

Хеш-функція має такі властивості:

а) не існує зворотної функції h', такої що $h'(y) = x$; частіше за все неможливо однозначно відновити елемент по його хешу;

б) припускають так звані колізії, тобто такі, що $h(x_1) = h(x_2) = y$; два абсолютно різні елементи можуть дати однаковий хеш;

Проблема колізій елементів має важливе значення при розв'язку задач, однак її можна побороти різними способами, в залежності від умов задачі. Найпоширенішим способом буде введення додаткової(вих) хеш функцій, які будуть мінімізувати колізії за рахунок розширення розмірності (простору) вихідного хешу. Однак більш довге хеш-значення означає більше місця для зберігання та більше обчислень, що вплине на продуктивність та вартість зберігання. Тому потрібно робити вибір, щоб знайти баланс між безпекою та вартістю.

1.3.1 Хешування для ANN

Хеш-функцію можна використати для аналізу подібності елементів. Для цього використовуються різні міри, наприклад, відстань Хемінга, яка побітово порівнює два хешу і рахує кількість бітів, які відрізняються (цю операцію можна представити як виключну диз'юнкцію XOR). Відстань Хемінга буде набувати значень від 0 до N, де 0 буде означати ідентичність хешів (ідентичність або схожість

елементів), а N - максимальну різність елементів (при чому N буде дорівнювати фіксованому розміру хешу, наприклад, 64 біта). Але не для всіх хеш функцій можна використати властивість подібності, оскільки це залежить від самої хеш-функції та чи повертає вона «схожі» хеші для схожих елементів.

У цій роботі зосередимося в основному на хешуванні, залежному від даних, де хеш-функція вивчається з урахуванням розподілу вхідних даних. Репрезентативні методи цієї категорії включають методи навчання хешування (англ. Learning To Hash (L2H)).

L2H — це набір методів хешування залежних від даних, які мають на меті вивчити компактне представлення, що зберігає подібність, із коротшими хеш-кодами таким чином, що подібні вхідні дані відображаються на «сусідні» двійкові хеш-коди.

Якщо умова схожості хешів таки здійснюється, то для таких елементів можна побудувати хеш-таблицю, де для деякого значення хешу може зберігатися деякі схожі елементи. При цьому проблема колізій може виникати і для схожих елементів, що буде свідчити про те, що ці елементи є або схожими, або ідентичними, або (теж ймовірно) абсолютно різними. Для одного хешу буде накопичуватися деяка множина елементів, що будуть між собою схожими. Всі ці елементи ланцюжком (списком) записуються в одну комірку хеш-таблиці. Але, якщо кількості елементів недостатньо для формування результату, наприклад, якщо визначено фіксовану кількість K-найближчих сусідів, то ми можемо подивитися в «сусідні» комірки, які відрізняються від заданого хешу на один біт. Наприклад, хеш 001, має наступних сусідів: 101, 011, 000. Вони будуть «сусідніми», бо відстань між заданим і наведеними хешами буде дорівнювати 1 (за Хемінгом) (див. Рисунок 1.3). Таких сусідів буде всього N - фіксований розмір хешу. За допомогою сформованої хеш-таблиці можна швидко знайти схожі елементи, складність такої операції буде залежати від реалізації власне структури хеш-таблиці, але в ідеальному випадку буде складати O(1).

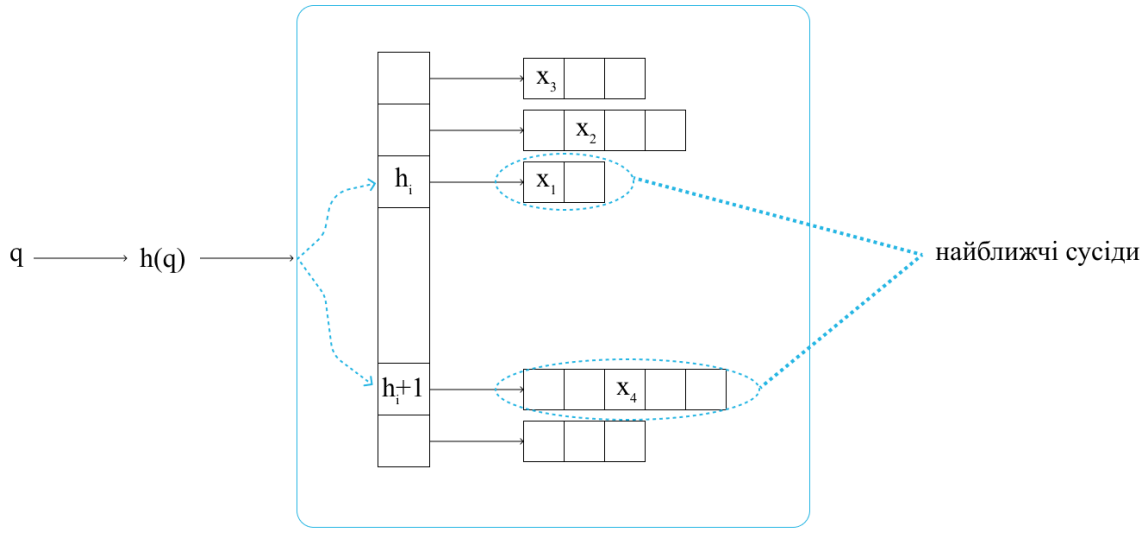


Рисунок 1.3 – Хеш-функція використовується для визначення найближчих сусідів, і визначає результат як елементи, які знаходяться у хеш-таблиці під хеш-кодом, який дорівнює хеш-коду запиту, або відрізняється на 1 біт від нього (відстань 1 за Хемінгом).

У такий спосіб ми значно зменшуємо та спрощуємо задачу, оскільки короткі двійкові хеші, які використовуються як зразкове (англ. proxy) представлення деякого елементу, значно легше порівнювати ніж багатовимірні елементи з реальними значеннями. Це значно зменшує витрати на обчислення відстані між елементами і покращує алгоритм пошуку як у часі, так і в просторі, тому що розмір структур даних, що необхідно обробити під час пошуку та кількість операцій порівняння між елементами значно зменшується.

1.4 Система пошуку зображень на основі вмісту

Система пошуку зображень на основі вмісту (англ. Content Based Image Retrieval (CBIR)) - це система, яка за даним запитом (наприклад, зображенням) пропонує схожі або ідентичні зображення з власної бази зображень. Цей метод повністю автоматизований, однак страждає від «семантичного розриву» [3], який є розривом між ознаками низького рівня, що описують зображення, і високорівневими концепціями (сприйняттям), що містяться в зображеннях, що призводить до

нерелевантного пошуку зображень. Дані системи зазвичай розпізнають так звані «глобальні» ознаки (наприклад, колір, текстура, форма) та «локальні» (специфічні особливості, ключові точки або деякі частини зображень, наприклад, кути, краплі та краї). Часто системи можуть спиратися на класифікацію об’єктів, що дозволяє їм ранжувати пошукові запити.

Загальна структура CBIR складається з деяких етапів. Перший етап - етап препроцесингу зображення, який застосовується до всіх зображень (зображень-запитів та зображень, які було попередньо додано до бази). До препроцесингу може входити зміна розміру зображення, ротація, виділення границь, перетворення в чорно-білі зображення, тощо.

Далі слідує етап вилучення ознак зображення. Необов’язковим етапом може бути класифікація та сегментація об’єктів на основі вилучених ознак. Останнім етапом є порівняння подібності між вилученими ознаками із зображення запиту та всіма іншими зображеннями в базі, щоб отримати найбільш релевантні зображення. Зворотній зв’язок щодо релевантності – це ще один можливий етап, який покращує результати завдяки втручанню користувача.

1.5 Нейронна мережа

Застосування нейронних мереж можна побачити майже в будь якій сфері, а їх розповсюдження стає подалі більшим і масштабним. Наразі важко уявити деякі тривіальні речі без існування нейронних мереж, та все ж таки нейронні мережі не панацеєю для усіх задач. Потенціал нейронних мереж все ще зростає, а приклади застосування ще доповнюються. Нейронні мережі тісно пов’язані з темою штучного інтелекту (англ. artificial intelligence) та машинного навчання (англ. machine learning).

Нейронні мережі навчаються на деякому наборі навчальних даних, щоб потім передбачати, кластерувати та/або класифікувати схожі дані. Процес цей ітеративний і продовжується до поки точність нейронної мережі збільшується. Але як тільки мережа сформована, вона стає потужною системою, яка дозволяє нам

робити передбачення з високою швидкістю. У свою чергу, аналіз, який проводять експерти, може займати хвилини або навіть години, що збільшує цінність таких інтелектуальних систем, і часто замінює людську працю.

Нейронна мережа дуже елегантна і в той же час проста математика. Загальний принцип роботи нейронної мережі заснований на прикладі біологічних нейронів.

Штучна нейронна мережа (Artificial Neural Network) містить велику кількість процесорних одиниць, які називаються перцептронами. Так само як нейрон за допомогою дендритів приймає вхідні імпульси, так і нейрона мережа приймає вхідні значення за допомогою перцептронів. Далі ці дані деяким образом обробляються і передаються в наступні перцептрони, а на виході ми отримаємо деяке передбачення (оброблену відповідь). Найпростіша нейронна мережа складається з одного перцептрона, який може приймати багато вхідних значень, і видавати деяку відповідь засновану на них. Стандартно, нейронні мережі складаються з великої кількості перцептронів, тому їх називають багатошаровими нейронними мережами (англ. Multilayer Perceptron Networks) [18].

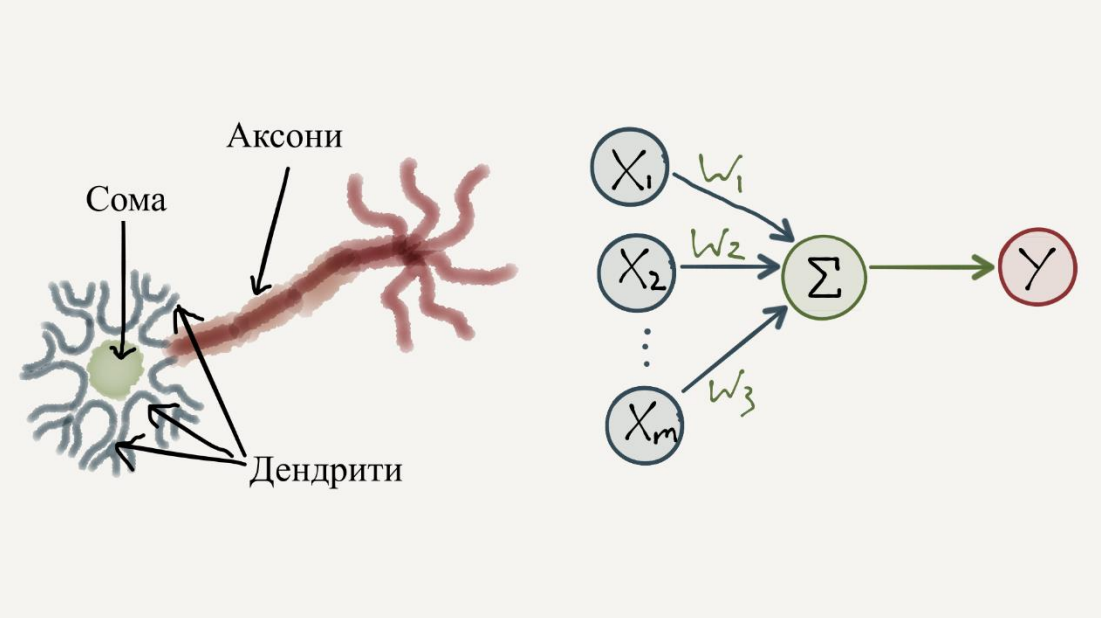


Рисунок 1.4 – Структура нейрона (зліва) та перцептрона (справа) [17].

Перцептрони з'єднані один з одним за допомогою зв'язку, який має деяку вагу. Ваги можна налаштувати так, щоб нейронна мережа могла правильно

обробляти інформацію, що схоже на адаптацію синапсу нейронів. Кожен перцептрон в мережі має активаційну функцію, яка визначає, чи потрібно передавати сигнал далі, що, по суті, є значенням толерантності перцептрона. Вихідне значення перцептрона можна визначити, як сума вхідних даних, помножена на ваги відповідного з'єднання. Уявімо, що у нас є перцептрон P з двома вхідними значеннями i_1 і i_2, які передаються по зв'язкам з вагами w_1 і w_2. Тоді, вихідне значення в перцептроні P буде дорівнювати $P = w_1 i_1 + w_2 i_2$.

По суті можна провести аналогію, що наш мозок є деякою абстракцією з вагами та порогами, яка приймає вхідні дані та обробляє їх за допомогою ваг, даючи нам певний результат.

Нейронна мережа складається з великої кількості перцептронів, які організовуються у шари. Вирізняють три види шарів - вхідні, вихідні та приховані. Приховані шари - це ті, що розташовуються між вхідним і вихідним шаром, і які відповідають безпосередньо на обробку вхідних даних. Чим більше шарів, тим складнішою стає система, що дозволяє їх робити більш складні прогнози, але робить мережу більш тяжкою для тренування.

Власне тренування мереж, коли ваги перцептронів підлаштовуються під певний тип даних, відбувається завдяки математичному методу, який називається градієнтний спуск (англ. Gradient Descent). Геометричне інтерпретування даного методу полягає в пошуку деякої точки в просторі яка буде мінімальною для деякого околу, і буде мінімізувати функцію втрат. Функція втрат - це деяка міра, яка дозволяє зрозуміти, наскільки гарно робить передбачення мережа на деякому наборі даних.

Більшість глибоких нейронних мереж мають прямий зв'язок (англ. feedforward). Дані проходять тільки в одному напрямку, починаючи з вхідного шару та закінчують вихідним. Проте, мережу можна навчити шляхом зворотного поширення (англ. backpropagation), тобто в зворотному напрямку, що дозволяє визначити помилку, пов'язану з кожним перцептроном та дозволить належним чином підлаштувати параметри мережі.

Вихідними даними нейронної мережі може бути будь-що, але зазвичай це деякий вектор, який класифікує вхідні дані. Це може бути або просто відповідь чи належать дані до деякого класу, або може бути і ймовірність приналежності даних до деякого класу. Мережа може передбачати і багато класів для одного вхідного об’єкту, або може строго видавати один клас.

Часто вводиться термін глибокого навчання (Deep Learning), що інколи замінює термін нейронних мереж. Варто зазначити, що навчання стає глибоким, коли нейронна мережа має багато шарів (частіше коли шарів більше трьох). Звідси і береться поняття «глибини» в назві глибинного навчання.

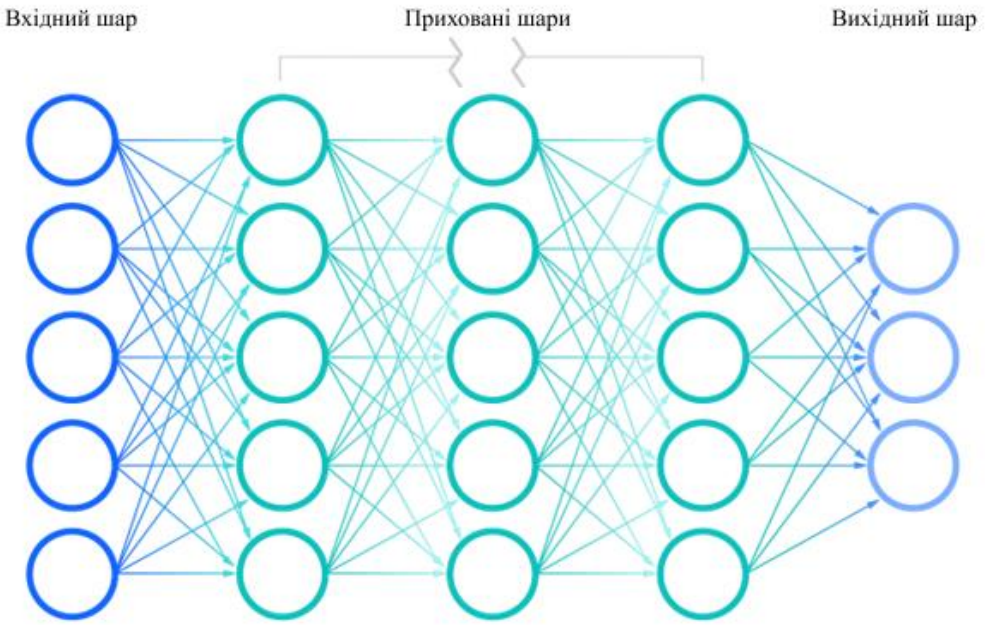


Рисунок 1.5 – Структура глибокої повнозв’язної нейронної мережі [22]

Більш складні нейронні мережі здатні приймати більш складні вхідні дані, наприклад зображення. Часто, коли мова йде про зображення, то спливають так звані згорткові нейронні мережі (англ. Convolutional Neural Networks (CNNs)), які використовують особливі згорткові шари, щоб зменшити розмірність даних [22].

Існують мережі, які засновані на зворотному зв’язку. Такі мережі називають рекурентними нейронними мережами (англ. Recurrent Neural Networks (RNNs)) і часто використовуються для прогнозування будь-чого у майбутньому (часозалежні

дані), наприклад прогнозування стану фондових ринків, або продаж деякої компанії.

Нейронні мережі можна також класифікувати за сімейством архітектури, яку вона використовує (наприклад, популярні U-мережі, які мають форму відповідної літери).

1.5.1 Функція втрат

Функція втрат (англ. loss function) - це метод оцінки, який визначає наскільки гарно мережа прогнозує на деякому наборі даних. Якщо прогноз повністю не співпадає з реальністю, то функція втрат виведе велике значення, і навпаки, якщо повністю співпадає, то значення буде 0. Власне функція втрат допомагає зрозуміти чи зміни під час тренування пішли на користь моделі чи ні.

Функції втрат пов’язана з точністю моделі, ключовою характеристикою машинного навчання. Існує безліч різних функцій втрат, найпопулярніші серед яких середньоквадратична похибка (англ. Mean squared error) та Перехресна ентропія (англ Log loss або cross entropy loss).

Задача ускладняється коли модель є мультикласовим класифікатором (видає декілька класів як передбачення). Розглянемо спосіб обчислення точності (англ. accuracy) для мультикласового передбачення. Оскільки нейронна мережа буде генерувати передбачення для кожного класу як деяке дійсне значення приналежності до цього класу, то першим кроком буде нормалізація цих значень. Зазвичай в нейронних мережах використовують сигмоїду - це функція, яка згладжує деякі значення і відображає їх на проміжок (0, 1). Визначається вона як

$$S(x) = \frac{1}{1+e^{-x}}.$$

Таку функцію, як сигмоїда, також називають функцією активації. Власне функції активації використовуються у нейронних мережах щоб відобразити вихідні значення шарів в деякі значення на проміжку [-1, 1] або [0, 1]. Після застосування сигмоїди до вихідних значень, треба відсіяти нерелевантні передбачення. Для цього можна використати деякий поріг (англ. threshold) - всі значення які більше

цього порогу будуть вважатися як індикатор приналежності до цього класу, а всі, що менше - будуть казати про неприналежність. Після цього, можна порівняти реальні класи з тими що передбачила мережа. Співвідношення класів, які співпали (істинно позитивні (true positive) та істино негативні (true negative)), до загальної кількості класів, і буде вважатися як мультикласова точність передбачення.

$$accuracy = \frac{TP+TN}{N},$$

де TP - істинно позитивні, тобто ті, які дійсно є такими класами,

TN - істинно негативні, тобто ті, які дійсно не відносяться до цих класів,

N - загальна кількість класів.

1.5.2 Темп навчання

Нейронні мережі навчаються за допомогою алгоритму оптимізації стохастичного градієнтного спуску. Стохастичний градієнтний спуск — це алгоритм оптимізації, який оцінює помилку для поточного стану моделі за допомогою прикладів із навчального набору даних, а потім оновлює ваги моделі за допомогою алгоритму зворотного поширення (backpropagation). Існує багато варіацій стохастичного градієнтного спуску, так звані оптимізатори: Adagrad, Adam, Adadelta тощо. Усі оптимізатори вимагають початкового налаштування темпу навчання. Значення, на яке мають оновитися ваги під час тренування, називають темпом навчання або «швидкістю навчання».

Темп навчання або коефіцієнт швидкості навчання (англ. learning rate) — це гіперпараметр, який контролює, наскільки змінювати модель у відповідь на значення помилки кожного разу, коли ваги моделі оновлюються. Обрати значення темпу навчання є нетривіальною задачею, оскільки занадто мале значення може призвести до тривалого процесу навчання, а занадто велике значення може призвести до нестабільного процесу навчання.

Темп навчання є одним з найважливішим гіперпараметром під час налаштування нейронної мережі. Тому варто дослідити значення темпу швидкості навчання до повноцінного тренування моделі.

Темп навчання — це настроюваний гіперпараметр, який має невелике додатне значення, часто в діапазоні від 0 до 1.

Тренування слід починати з відносно великої швидкості навчання, оскільки на початку випадкові ваги далекі від оптимальних, а потім швидкість навчання може зменшуватися під час навчання, щоб дозволити більш тонко оновлювати ваги моделі.

Існують різні методи для підбору значень для темпу навчання. Наївний підхід полягає в тому, щоб спробувати кілька різних значень і побачити, яке з них принесе найкращі втрати, не жертвуючи швидкістю навчання, наприклад, почати з великого значення, припустимо 0.1, а потім спробувати експоненціально менші значення: 0.01, 0.001 тощо. Далі, спостерігаючи за значеннями функції втрат (англ. loss function), можна зробити висновок чи підходить це значення, чи варто його зменшити.

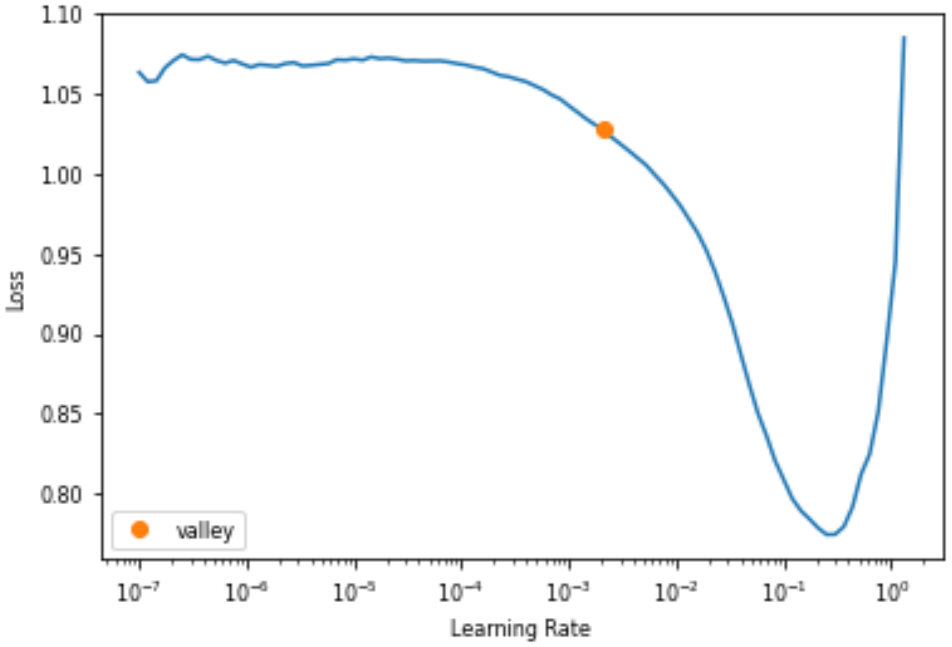


Рисунок 1.6 – Графік залежності значень функції втрат від значень темпу навчання.

Більш «розумний» метод підбору темп навчання описав Леслі Н. Сміт в своїй роботі [14]. Метод полягає в тому, щоб навчати мережу, починаючи з низького значення темпу, і збільшувати його експоненціально для кожної партії (англ. batch).

Записавши втрати навчання для кожного такого значення можна побудувати графік залежності втрат від темпу навчання. Отримаємо графік схожий на Рис 1.6. Втрати спочатку зменшуються, потім тренувальний процес починає розходитися. На графіку потрібно вибрати точку, де втрати найшвидше зменшуються. У цьому прикладі значення втрат швидко зменшується, коли швидкість навчання становить від 10^{-2} до 10^{-3}.

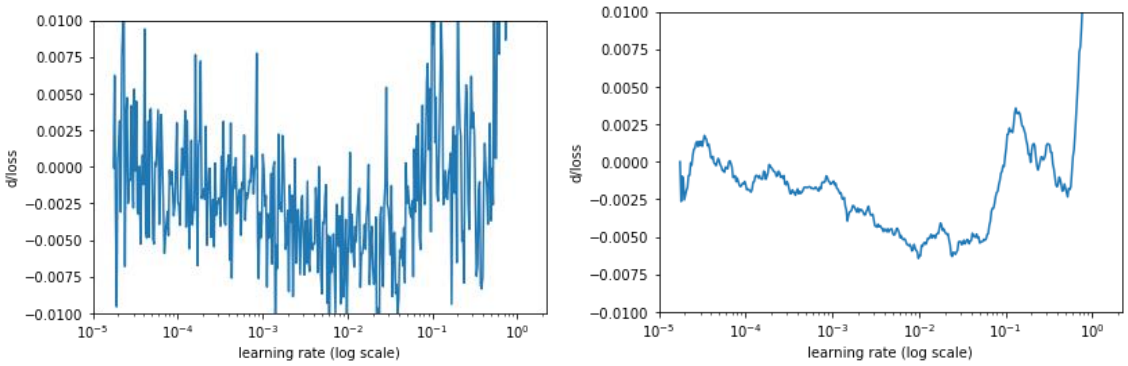


Рисунок 1.7 – Графік похідної функції втрат без згладження (зліва) та зі згладженням (справа) [16].

Інший спосіб — розрахувати швидкість зміни втрат (похідна функції втрат), потім побудувати графік. Отриманий графік буде виглядати дуже зашумленим, тому потрібно додатково застосувати згладження, наприклад, середнє в деякому околі. Після цього графік стане більш стабільним. Мінімальне значення на побудованому графіку буде давати найкращий темп навчання. Таке значення називають valley (перек. межигір'я або долина) що видно з Рис 1.7.

1.5.3 Оптимізатор Adam

Градієнтний спуск є одним із найпопулярніших алгоритмів для оптимізації та найпоширенішим способом оптимізації нейронних мереж. Часто, реалізація цього алгоритму вже вбудована в сучасні фреймворки і не потребує від розробника зусиль на імплементацію. Розглянемо найвідоміший та найбільш вживаний оптимізатор Adam.

Адаптивна оцінка моменту (англ. Adaptive Moment Estimation) це метод, який використовує адаптивний темп навчання для кожного параметра. Адам спирається на фізичному підґрунті - симулює важку кулю, що котиться по схилу, враховуючи тертя. Таким чином, куля знаходить мінімуми на поверхні що описується функцією втрат. Автори оригінальної статті емпірично показують, що Адам добре працює на практиці і вигідно відрізняється від інших алгоритмів адаптивного методу навчання.

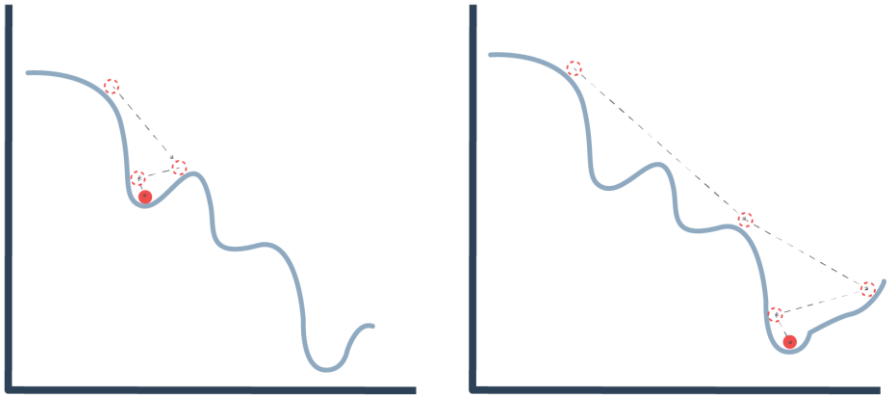


Рисунок 1.8 – Приклад роботи Adam при маленькому початковому темпі навчання (зліва) та більшому (справа) [23].

1.6 Згорткова нейронна мережа

Зазвичай складні вхідні дані, такі як зображення, треба попередньо обробити та спростити, щоб звести задачу до більш простої. Такий підхід використовують згорткові нейронні мережі (англ. Convolutional Neural Network (CNN)), і присутні в них згорткові шари (англ. convolutional layers (Conv)). Такі шари зменшують розмірність вхідних даних, і намагаються витягти з них найбільш релевантну інформацію. Назва пішла від латинського слова convolvere, що означає згортатися. Різні рівні згорткових мереж можуть витягувати різні типи ознак, починаючи з найпростіших, такі як колір та лінії, продовжуючи більш складними, такі як форма і текстура, і закінчуючи комплексними, такими як розпізнавання цілих об'єктів. Основна ціль згорткових шарів - виділити певну абстракцію для конкретних ознак, і на практиці такі «абстрактні» ознаки дуже важко інтерпретувати.

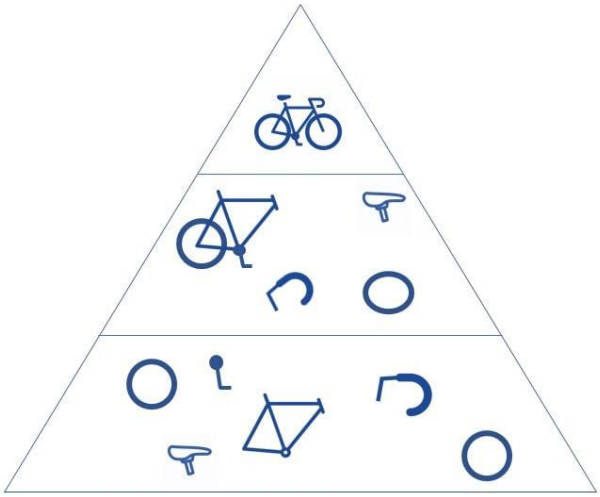


Рисунок 1.9 – Абстрактне пояснення роботи нейронної мережі – нижні рівні зображають роботу згорткових шарів на початку мережі, а верхні – на кінці [21].

Згорткові мережі частіше за все використовуються для аналізу зображень та в основному для класифікації зображень, групування їх за подібністю (пошук зображень) і розпізнавання об’єктів. Великою перевагою таких мереж є просторова адаптивність, тобто незалежність розміщення об’єкта у просторі, що значно покращує результати пошуку та класифікації об’єктів незалежно від їх розміщення.

Згорткові мережі складаються з декількох видів шарів. Найзначущі шари - це згорткові шари. Принцип роботи такого шара заключений у так званому ядрі згортки (англ. kernel), яке ще називають фільтр (англ. filter) - це невелика матриця з деякими коефіцієнтами, які налаштовуються під час навчання мережі. Ядро обробляє вхідне зображення наступним чином: до кожного фрагменту застосовується по елементне множення, а потім отримані результати сумуються. Цей принцип дуже схожий на метод розсувного вікна (англ. sliding window kernel), та суттєво зменшує розмінність даних.

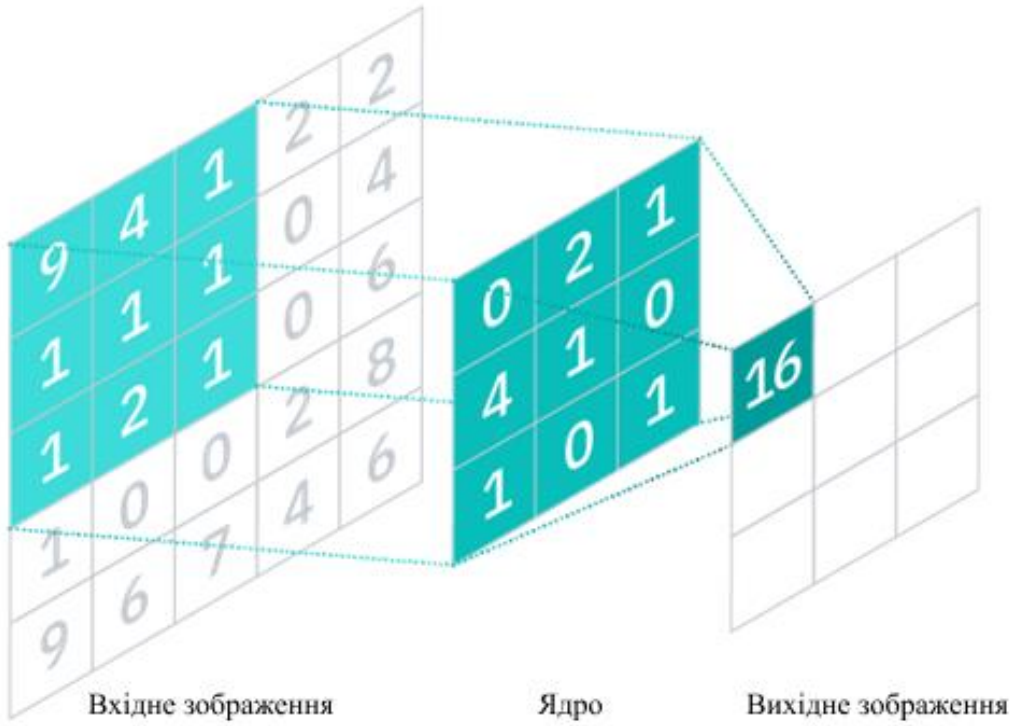


Рисунок 1.10 – Приклад роботи згорткового шару, де використовується ядро (англ. filter) [21].

Після згорткових шарів йдуть шари активації (функції активації). Частіше за все використовують функцію активації, яка називається ReLU (англ. rectified linear unit), яка визначається як $f(x) = max(0, x)$. У такий спосіб відсіюються негативні значення, і залишаються тільки додатні значення скалярних величин. Основне призначення таких шарів - відфільтрувати нерелевантні дані, і зосередитися на значущих ознаках, які були вилучені за допомогою згорткових шарів.

Наступним шаром є шар пулінгу (pooling), який ще називають шаром субдискретизації. Цей шар ущільнює дані, де група деяких пікселів ущільнюється до одного. Наприклад, можуть використовувати групи 2 на 2, що відповідно зменшить зображення у два рази. Щоб ущільнити групу використовують функцію субдискретизації, але частіше за все у ролі такої виступає функція max. Шар пулінгу так само зменшує розмір вхідних даних, і основна його ідея полягає у тому, що якщо на попередньому рівні вже були виявлені деякі ознаки, то далі настільки детальне зображення вже не потрібно, тому треба його зменшити до більш простого. Такий шар частіше за все вставляється перед наступним згортковим шаром.

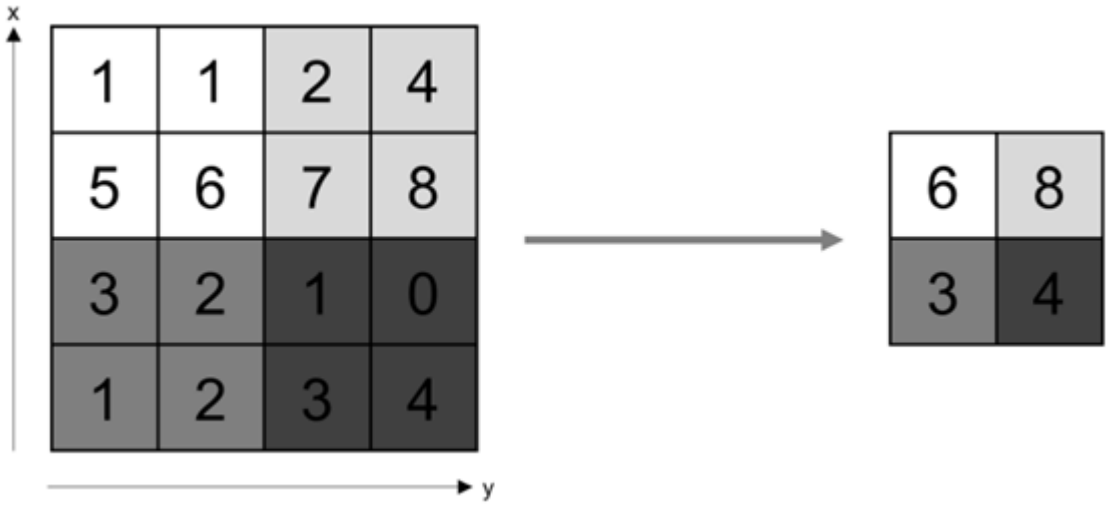


Рисунок 1.11 – Приклад роботи пулінг (англ. max pooling) шару [5].

Після того, як дані пройшли декілька рівнів згорткових шарів, вони готові для стандартної архітектури нейронної мережі, а саме до повнозв’язних шарів (англ. Fully Connected Layers). В класичних мережах використовують саме такі шари, де кожен вузол зв'язаний з кожним вузлом попереднього рівня. На відміну до повнозв’язних шарів, згорткові шари з’єднують тільки деяку кількість попередніх вузлів з наступним (якщо ядро розміром 3 на 3, то тільки 9 вузлів зв’язані з наступним). Такі шари відіграють роль класифікатора в мережі - власне вони намагаються визначити комбінацію ознак, вилучених на етапі згорткових шарів, що належать деякому класу.

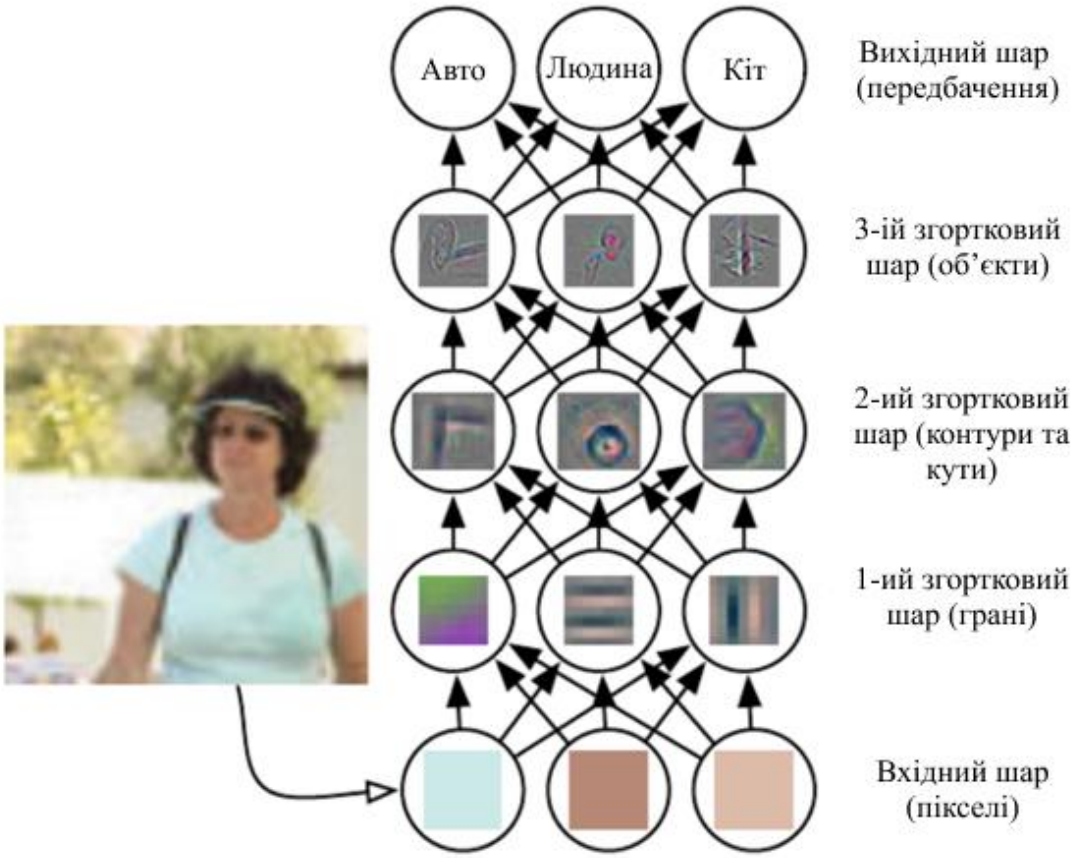


Рисунок 1.12 – Приклад того, як згорткова нейронна мережа вилучає ознаки зображення з розбивкою по шарам [4].

Наразі найбільш популярні архітектури згорткових мереж відносяться до таких сімейств: ResNet, VGGNet, LeNet, GoogLeNet, AlexNet.

Згорткові нейронні мережі є потужним інструментом комп'ютерного зору. Застосування згорткових нейронних мереж можна побачити у таких сферах, як маркетинг, медицина, торгівля, транспорт, космонавтика тощо. Наприклад, деякі соціальні мережі розпізнають обличчя людей на фото, що полегшує відмічати друзів на них. Часто згорткові мережі використовуються для радіологічних досліджень, щоб допомагає лікарям швидко знаходити та ідентифікувати ракові пухлини на знімках. У роздрібна торгівлі мережі використовують для систем рекомендацій, що дозволяє деяким брендам одягу рекомендувати речі, які доповнюють наявний гардероб. Найпопулярнішим прикладом застосування мереж є самокеруючі автівки. Безпілотні автомобілі вже почали з'являтися на дорогах, однак застосування згортковим мережам можна знайти не тільки у керуванні автомобілем, а й розпізнаванні

дорожніх знаків, інших автомобілів, перешкод тощо, що у сукупності допомагає підвищити безпеку керування автомобілем людиною.

Але незважаючи на потужність і складність згорткових нейронних мереж, вони не позбавлені недоліків. CNN дуже гарно справляються з пошуком ознак, які часто непомітні для людського ока. Але коли мова йде про визначення змісту (семантики) зображення, то мережа не завжди може впоратися з задачею, оскільки оперує лише патернами, які дуже складно передбачити для усіх сценаріїв.

1.7 Види нейронного навчання

Існує безліч різних способів і методів навчання в сфері глибокого навчання. Деякі способи є широко застосованими і використовуються майже в кожній задачі нейронного навчання (наприклад, трансферне навчання), а є більш екзотичні методи, які тільки набувають популярності (наприклад, онлайн навчання).

1.7.1 Трансферне навчання

Для вирішення більшості задач машинного навчання використовують спільні кроки та методи. Наприклад, крок вилучення ознак з зображення присутній майже у всіх задачах, які пов’язані з комп’ютерним зором. Звичайно це вимагало б від дослідника створення власної нейронної мережі та тренування на великій кількості зображень, щоб та гарно могла вирізняти ознаки та особливості зображень. Але існує метод, який здобув назву трансферне навчання (англ. Transfer Learning), який дозволяє досліднику використовувати вже натреновану модель, щоб створити власну. Припустимо ми маємо вже готову мережу, яка гарно вирізняє деякі ознаки на зображені, тоді ми можемо використати її, і додати відповідні повнозв’язні шари (шари для класифікації), і дотренувати модель на власних даних і класах. Таким чином ми економимо час на тренування згорткових шарів, і нам знадобиться небагато ітерацій щоб отримати готову модель. Традиційні моделі машинного навчання у свою чергу вимагають навчання з нуля, що є витратним з точки зору обчислень і вимагає великої кількості даних для досягнення високої продуктивності. Також,

претреновані мережі вже оцінені і протестовані, що дає впевненість, що вони будуть ефективно працювати. Ще однією перевагою є те, що можна обрати мережу відповідно до її розміру і кількістю шарів, що може бути вимогою залежності від задачі і пристрою, де буде розгорнуто модель.

«Трансферне навчання … відносяться до ситуації, коли те, що було вивчене в одній обстановці … використовується для покращення в іншій обстановці.» [9] «Трансферне навчання – це вдосконалення навчання в новому завданні шляхом передачі знань із суміжного завдання, яке вже засвоєно.» [6]

Трансферне навчання є популярним методом машинного навчання, оскільки не вимагає величезних ресурсів та даних, на яких відбувається тренування, для того щоб навчити мережу.

Варто зазначити, що всі моделі, які підготовлені для трансферного навчання, натреновані, щоб визначати узагальнені ознаки та особливості даних. Щоб підготувати таку модель спочатку треба навчити базову мережу на деякому простому наборі даних, щоб потім можна було змінити профіль цільової мережі, додавши і натренувавши на власному наборі даних.

Часто, у вирішені нових задач комп’ютерного зору використовують такі мережі, як ResNet50 та VGG16.

1.7.2 Адаптивне навчання

В умовах, коли набір даних не може передбачити всі класи або випадки, на допомогу можу прийти такий метод тренування мереж, як адаптивне навчання, або також зустрічаються назви онлайн (англ. online) або інкрементальне (англ. incremental) навчання. Адаптивне машинне навчання — це більш просунуте рішення, яке серйозно ставиться до збору й аналізу даних у реальному часі. Такий метод адаптується до нових даних на льоту і вилучає з них необхідні ознаки і особливості, що допомагають сформувати деяке представлення для класифікації, або будь якої іншої задачі. Адаптивне навчання може нескінченно збирати нові дані, аналізувати їх і поповнювати свої знання. Тому модель постійно покращується і

робить більш точні прогнози для майбутніх даних. Ще одна перевага такої мережі є те, що система не ризикує стати застарілою, що зменшує витрати на постійне оновлення мережі.

Як приклад застосування такої мережі, можна привести фінансовий сектор – де для прогнозування ринку, потрібно постійно аналізувати нові тенденції. Ще одним прикладом може слугувати медицина, де таку мережу можна використати для точної та доступної діагностики, та попередження про проблеми до того, як вони виникнуть.

Адаптивне або онлайн навчання — це один з найпотужніших методів, яке нещодавно з’явилася у дослідженнях і в промисловості. Метод може використовуватися у різних сценаріях та налаштований на постійну зміну завдяки збору та обробки нових даних.

1.7.3 Активне навчання

При традиційному навчанні нейронних мереж збирається велика кількість даних і попередньо обробляється, наприклад проставляються мітки або виділяються об’єкти. Після досягнення достатньо великої кількості тестових даних тренується мережа. Однак для складних задач, особливо які вимагають велику кількість класів та випадків, дуже складно зібрати таку велику базу даних. Це є однією з найбільш трудомістких завдань у пасивному (традиційному) навчанні. Тому часто застосовують так зване активне навчання (англ. active learning). Основна гіпотеза активного навчання полягає в тому, що під час навчання нейронна мережа може обирати найбільш ефективні дані, які наблизять її до розпізнавання або вирішення конкретної задачі.

Активне навчання — це метод, який використовується у тренуванні нейронних мереж, де навчання відбувається у інтерактивний спосіб, і мережа обирає дані які вважає найбільш релевантними для навчання. Часто це може супроводжуватися запитами до реального користувача, щоб промаркувати деякі дані і включити їх до навчальних даних. Прикладом такого навчання є Captcha - система для

розпізнавання користувача, яка використовує зображення тексту, номерних знаків, велосипедів, світлофорів, автобусів тощо, та людську роботу для того щоб промаркувати дані і навчитися на них розпізнавати ці об’єкти.

Під час навчання за допомогою активного методу мережа вибирає деяку множину даних-екземплярів, які мають найбільший інтерес. Потенційно мережа може досягти більш високого рівня точності, використовуючи меншу кількість даних, оскільки їй дозволяється обирати дані, на яких вона хоче тренуватися.

Часто активне навчання порівнюють з іншим підходом машинного навчання, а саме з «навчанням з підкріпленням» (англ. reinforcement learning). Навчання з підкріпленням і активне навчання можуть зменшувати необхідну кількість даних для тренування, але це різні поняття. Головна відмінність у тому, що навчання з підкріпленням цілеспрямований метод, який тренується на власних помилках, і напряму взаємодіє з навколишнім середовищем. Мережа навчається методом спроб і помилок, і отримує визначену винагороду за правильні рішення, і штрафи - за неправильні. Також навчання з підкріпленням часто не потребує збирання даних, тому що він генерує власні під час тренування.

Активне навчання все ж таки відноситься до традиційного навчання з вчителем. Його можна назвати навчанням з напівнаглядом, тобто модель навчаються як на промаркованих даних, так і непромаркованих. Основна складність полягає в тому, щоб визначити які дані принесуть найбільшу користь під час навчання.

1.8 N-кадрове навчання

Задачі машинного навчання часто стикаються з проблемою, коли навчальних даних недостатньо, щоб натренувати гарну мережу. Тому було розроблено схожі алгоритми, які назвали N-кадровим навчанням, інколи навчання N-пострілів (англ. N-shot learning). N-shot навчання – це спосіб навчання нейронних мереж, коли використовується невелика кількість навчальних даних. Навчання N-shot використовує N зразків для кожного класу K, а отже весь набір даних включає $S = N \times K$ зразків. Є три основних підвидів такого навчання: нуль-кадрове, одно-кадрове та

кілька-кадрове навчання. Який саме алгоритм використовувати, залежить від наявності навчальних даних. N-shot можна використовувати коли величезна кількість даних, що накладає обмеження у часі і вартості його маркування.

1.8.1 Нуль-кадрове навчання

Основна ідея, яка описує нуль-кадрове навчання, це класифікація об’єктів із раніше небачених класів. Zero-Shot Learning (ZSL, нуль-кадрове, нуль-пострілів навчання) — це парадигма машинного навчання, де попередньо навчена мережа створюється для категоризації нових зразків, класи яких не присутні в навчальному наборі. Наприклад, візьмемо мережу, яка вміє розпізнавати зображення котів і собак, і навчимо її розпізнавати зображення птахів. Класи «кіт» і «собака» а даному випадку будуть називатися «побаченими» класами, тобто ті, які мережа вже знає і бачила, а нові навчальні екземпляри називаються «небаченими».

Нуль-кадрове навчання має спільні ідеї з трансферним навчанням, і по суті є його підмножиною. Найпоширеніша форма трансферного навчання - це «гомогенне трансферне навчання», коли попередньо натреновані мережі використовують схожі дані. Однак, нуль-кадрове навчання належить до «гетерогенного трансферного навчання», бачені і небачені екземпляри часто несумісні.

Нуль-кадрове навчання імітує властивість людини, яка може описати будь який предмет, навіть якщо бачить його в перший раз. Це можливо завдяки наявній базі слів (класів), значення яких людина легко переносить на інші предмети, описуючи їх особливості. Люди, природно, можуть знаходити подібність між класами даних. Наприклад людина з легкістю знаходить такі спільні ознаки у кота і собаки - мають хвости, обидва ходять на чотирьох тощо [19].

У нуль-кадровому навчанні видимі і невидимі класи представлені у багатовимірному векторному просторі, який називається семантичним простором, де знання з видимих класів можуть бути передані в невидимі класи.

Прикладом нуль-кадрового навчання є і алгоритм нейронного перенесення стилю (Neural Style Transfer), коли мережа може використовувати будь-яке

зображення (яка раніше не бачила) як зразок, що застосувати стиль на інше. Також нуль-кадрове навчання використовується в обробці природної мови (англ. Natural Language Processing), при пошуку зображень (Image Retrieval), для розпізнавання об’єктів (англ. Object detection) та для семантичній сегментації (англ. Semantic segmentation).

1.8.2 Одно-кадрове навчання

Одно-кадрове навчання (англ. One-shot learning) використовує зразкові дані для категоризації або розпізнавання об’єктів. Найпопулярніше застосування одно-кадровому навчанню можна знайти в аеропортах, де на автоматичних пропускних системах (паспортний контроль), даний тип нейронних мереж розпізнає людей по обличчю, співставляючи їх з фото в паспорті. Власне особливість використання одного «зразка» вирізняє одно-кадрове навчання від нуль-кадрового, всі інші особливості даних мереж пересікаються.

1.8.3 Кілька-кадрове навчання

Кілька-кадрове навчання (англ. Few-shot learning), також відоме як низько-кадрове навчання (англ. Low-shot learning), використовує невелику кількість зразків, для визначення нового класу. Використовуючи попередні знання, кілька-кадрове навчання може швидко узагальнити нові завдання, які містять лише кілька зразків із контрольованою інформацією [7]. Різниця між одно-кадровим і кілька-кадровим навчанням можна побачити у назві, тобто кілька-кадрове навчання дозволяє більшу кількість зразків, що правда цієї кількості не достатньо для задач традиційного машинного навчання.

Розділ 2. ПРАКТИЧНІ АСПЕКТИ АЛГОРИТМУ

2.1 Багатокласова класифікація

Багатокласова (багатоміткова) класифікація (англ. Multi-label classification (MLC)) - популярна задача для нейронних мереж, яка передбачає прогнозування декількох класів (міток) для одного запиту. Прикладом багатокласової класифікації є жанри книжок, які часто відповідають декільком жанрам одночасно (роман-хорор або комедія-драма тощо). Звичайно класів може бути скільки завгодно, обмеження тільки у кількості класів. Зазвичай нейронна мережа прогнозує належність тільки до одного класу, але зробити так, щоб вона прогнозувала відразу декілька досить тривіально. Оскільки останній шар нейронної мережі частіше за все прогнозує наскільки ймовірно належить запит до кожного класу, то можна відфільтрувати ті класи, які не перевищують деякий поріг, а ті класи, які залишаться, будуть і визначати багатокласовий результат.

Задачі багатокласової класифікації можуть відноситися до таких, де класи не є взаємовиключними, тобто можуть перетинатися. Однак і можуть бути взаємовиключні класи, що може бути у випадку коли на одному зображені будуть одночасно два класи.

2.2 Вилучення ознак зображення

Коли хешування застосовується для зображень, то важко знайти функцію таку, щоб перетворювала схожі зображення в схожі між собою хеші, оскільки важливу роль тут грає не значення пікселів у відповідному місці зображення, а власне зміст зображення, його семантична інформація. Тому, перед тим як застосовувати хеш функцію водять проміжний етап - вилучення (генерація) характеристик зображення, витягування з зображення смислового змісту, корисних ознак, а нерелевантна інформація нехтується.

Вилучення ознак — це процес зменшення розмірності даних, за допомогою якого початковий набір необроблених даних зводиться до більш компактних груп

для обробки. Характерною рисою цих великих наборів даних є велика кількість змінних, для обробки яких потрібно багато обчислювальних ресурсів. Вилучення ознак — це назва методів, які вибирають та/або об’єднують змінні в ознаки, ефективно зменшуючи обсяг даних, які необхідно обробити, при цьому точно й повністю описуючи вихідний набір даних. Вилучення ознак також може зменшити кількість зайвих даних для даного аналізу. Крім того, скорочення даних сприяє збільшенню швидкості навчання та зменшення кроків (епох) в процесі машинного навчання.

Після етапу вилучення ознак, згенеровані вектори ознак зображення проходять через хеш-функцію і перетворюються на компактні хеш-коди. При такому перетворені характеристики (ознаки) зображення зберігаються в деякій формі у згенерованих хеш-кодах.

Часто для генерації характеристик зображення використовують дескриптори, найпопулярніші з яких SURF, ORB, SIFT, або згорткові нейронні мережі, які будуть використовуватися в даній роботі. Добре відомо, що згорткові нейронні мережі вивчають абстрактні особливості на зображенні та вивчають ієрархію семантичних уявлень для зображень.

У той час як нижні шари нейронної мережі спеціалізуються на виявленні таких елементів, як границі або прості форми об’єктів, більш глибокі (високі) шари вивчають семантику змісту зображення, тобто відповідають за виявлення конкретних об’єктів на зображеннях.

Таким чином, відстань між згенерованими векторами-значеннями вищого рівня нейронної згорткової мережі можуть служити мірою подібності

зображень (семантично схожі зображення будуть генерувати схожі значення на відповідних вузлах деякого високого шару згорткової мережі).

Такий принцип використовується у нейронних мережах для перенесення стилю зображення (Neural Style Transfer), де мережу тренують таким чином, щоб зміст зображення зберігався незмінним, за допомогою вимірювання похибки подібності векторів ознак зображень (зі стилем і без) на деякому високому шарі заздалегідь натренованої згорткової мережі.

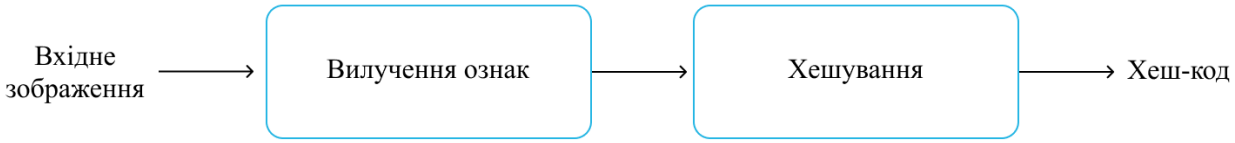


Рисунок 2.1 – Процес глибокого хешування (англ. Deep Hashing).

Використання нейронної мережі для пошуку подібних зображень можна побачити у роботі компанії Apple, яка використовує так звану NeuralHash (згорткову нейрону мережу) для визначення зображень, які містять CSAM (англ. Child sexual abuse material). Однак ця робота так і не була розгорнута на пристроях Apple, оскільки зазнало великої критики від спільноти через недоліки алгоритму, які буде розглянуто далі. Також критики зазнало те, що при досягненні певного порогу по виявленому CSAM зображень, працівники Apple отримували можливість на перегляд таких зображень. Але оскільки алгоритм міг помилково виявляти інший контент, це зазнало обурення у спільноти. В інших роботах можна побачити іншу назву розглянутого алгоритму, а саме Deep Hashing. Власне такої назви будемо притримуватись в цій роботі. Глибоке хешування являє собою контрольований L2H з використанням глибокого навчання (нейронних мереж) і включає в себе різні методи хешування, наприклад, ті що стосуються пошуку схожості в даних зображення. Зауважимо, що Deep Hashing може бути як суцільною мережею (тобто частина про хешування може бути вбудована як окремий шар/и в мережі) або просто як послідовний алгоритм з композиційних частин.

2.3 Позиційно чутливе хешування

Позиційно чутливе хешування (англ. Locality Sensitive Hashing (LSH)) - це метод хешування, який відображає деякі точки даних у сегменти, так що точки, які знаходяться поруч один з одним, розташовуються в одних і тих самих сегментах з високою ймовірністю, тоді як точки, віддалені один від одного, швидше за все, знаходяться в різних сегментах. Це полегшує пошук подібних елементів. Варто зазначити, що LSH відходить від стандартного класу функцій хешування тим, що

максимізує ймовірність колізій. Таким чином LSH намагається зменшити просторову розмірність особливостей в більш компактний хеш, за яким можна швидко знайти схожі елементи. Основний принцип LSH, це генерування випадковим чином деякої кількості гіперплощин, і потім визначення з якого боку знаходиться точка-елемент. Для кожної такої перевірки генерується двійкове значення (нуль або один) та формується вектор. За цим вектором легко визначити наскільки подібні вектори (і теоретично подібність елементів, з яких було отримано ці вектори) за допомогою різних метрик, наприклад відстані Хемінга.

Цей метод отримав назву випадкова проекція — техніка представлення багатовимірних даних у просторі ознак низької розмірності (зменшення розмірності). Він набув популярності завдяки своїй здатності приблизно зберігати відношення (попарна відстань або косинусна подібність) у маловимірному просторі, будучи менш дорогим з точки зору обчислень.

Основна ідея випадкової проекції полягає в тому, що якщо точки належать до багатовимірного векторного простору, то їх можна спроектувати у відповідний простір меншої розмірності таким чином, що приблизно зберігаються відстані між точками. Наведене вище твердження є інтерпретацією леми Джонсона-Лінденштрауса [11].

При використанні випадкової проекції, є ймовірність, що елементи попадуть до різних гіперплощин, хоча і будуть схожими. Тому щоб подолати такий недолік, можна зробити декілька різних випадкових проекцій. Схожі елементи з більшою ймовірністю попадуть в спільний сегмент хоча б в якійсь із проекцій. Це можна використати як деяку міру подібності - чим більше проекцій вважають такий елемент сусіднім - тим більше він схожий на запит.

Варто зазначити що LSH відноситься до методів хешування L2H (див Розділ 1). Як альтернативу для LSH можна навести KD-дерева, однак через складність збереження і більшу складність запитів, LSH можна вважати кращим алгоритмом для задачі пошуку найближчих сусідів (ANN). Сфери застосування LSH можуть включати рекомендаційні системи, виявлення майже дублікатів, ієрархічна кластеризація, ідентифікація схожості зображень (VisualRank), ідентифікація подібності звуку

(Shazam), цифровий відбиток відео тощо. Uber використовував LSH для виявлення зловживань платформою (фейкові облікові записи, шахрайство з платежами тощо). За допомогою LSH можна створити систему рекомендацій схожу на Youtube.

2.4 Атаки на нейронну мережу

Нейронні мережі, як будь-яка інформаційна система, схильна до різноманітних атак. Існують різні класифікації атак. Класифікуються вони залежно від того коли вони застосовуються: під час, до або після тренування мережі. Також класифікуються атаки відповідно до інформації, яку вони розкривають, наприклад, тренувальні дані (навчальний набір). Розберемо деякі потенційні атаки, які може зазнати наша мережа:

а) Радіоактивні дані (англ. Radioactive data) - тип атаки, яка визначає чи використовувався певний набір даних для навчання моделі. Радіоактивними вони називаються тому що в такі дані вносяться непомітні зміни, так що мережа, навчена на таких даних, буде мати ідентифікаційний знак. Знак стійкий до сильних варіацій, таких як різні архітектури мережі або методи оптимізації. Таким чином можна визначити чи використовувалися такі дані при навчанні. Така атака використовується на етапі навчання, і її важко виявити, оскільки важко перебрати весь набір навчальних даних, особливо коли береться вже готовий набір.

б) Крадіжка гіперпараметрів (англ. Hyperparameter theft) - тип атаки, коли зловмисник краде гіперпараметри моделі, щоб відтворити оригінальну модель. Різні гіперпараметри призводять до різної ефективності мережі – і часто стають комерційною таємницею. Така атака відбувається частіше на етапі тренування.

в) Інверсія моделі (англ. Model Inversion) або “висновок про членство” (англ. membership inference) - це такі типи атак, які намагаються відновити навчальний набір даних деякої моделі. Відбувається цей тип атаки після тренування, тобто коли модель вже використовується кінцевими користувачами.

Фредріксон, Джа і Рістенпарт продемонстрували метод виявлення даних, на яких була навчена модель, використовуючи той факт, що навчальні дані залишають свого роду відбиток у вагах нейронної мережі.

г) Змагальні атаки (англ. Adversarial attacks) - це тип атак, коли вхідні дані навмисно змінюються, для того щоб нейронна мережа не змогла їх класифікувати (або зробити по ним вірне передбачення).

Коли ми говоримо про радіоактивні дані, інверсію моделі та «висновок про членство», то варто брати до уваги навчальні дані, оскільки вони можуть стати доступні зловмисникам. Варто використовувати ті набори даних, які є доступні всім, або використовувати методи захисту, щоб не допустити такі атаки. Інакше, персональні дані користувачів можуть бути вилучені з самої мережі.

Також, всі атаки діляться на два типи - атака типу «білий ящик» та «чорний ящик». Атаки типу «білий ящик» характеризується тим, що хакер не має доступ до параметрів моделі, а ось атаки типу «чорний ящик» - хакер не має доступу.

2.4.1 Змагальні атаки

Змагальні атаки (Adversarial attacks) - метод, який намагається обдурити моделі машинного навчання за допомогою оманливих даних. Змагальна атака використовує навмисно розроблені дані, частіше навмисно видозмінені, задля того, щоб обдурити вже навчену модель.

Комісії національної безпеки США зі штучного інтелекту у 2019 році опублікувало звіт у якому зазначалося, що дуже невеликий відсоток поточних нейронних мереж спрямовані на захист від змагальних атак. Частина систем, які вже використовуються у виробництві є вразливими до таких атак. Наприклад, дослідники показали, що розмістивши на землі кілька маленьких наклейок, можна змусити самокерований автомобіль виїхати на зустрічну смугу руху. Інші дослідження показали, що систему медичного аналізу можна обдурити, якщо внести непомітні зміни у зображення, щоб вона класифікувала доброякісні родимки як злоякісні. Ще одним прикладом є шматочки наклеєної стрічки, які обдурили систему

комп’ютерного зору, щоб та неправильно класифікувала знак зупинки як знак обмеження швидкості.

Ймовірно, що зі збільшенням кількості систем штучного інтелекту, буде зростати і кількість змагальних атак, тому варто шукати способи як боротися з ними. Наразі це не тривіальна задача і вимагає багато досліджень.

Змагальні атаки можуть спричинити неправильну роботу нейронної мережі, коли підправлення деяким способом дані класифікувалися як “безпечні”. Це порушує усю цілісність системи і відводить її від поставленої цілі. Ще гірше, коли така атака відбувається на етапі тренування (наприклад під час активного або федеративного навчання), що дозволяє хакеру робити так звані бекдори (лазівку), щоб в майбутньому обдурювати модель. Таким чином зловмисник отруює дані впроваджуючи шкідливі зразки, які згодом порушують процес перенавчання. Внесені дані будуть помилково позначені як не шкідливі, але насправді будуть такими.

Змагальні атаки ще називають атаками ухилення, тобто такі, що дані змінюються, щоб уникнути виявлення або класифікуватися. Ухилення не передбачає впливу на дані, які використовуються для навчання моделі, але це можна порівняти з тим, як спамери та хакери приховують вміст спам-листів і шкідливих програм.

Вже згаданий приклад мережі NeuralHash, яка була запропонована Apple для пошуку CSAM контенту, була схильна до змагальних атак. Спільнота розробила спеціальний застосунок, який робив колізії абсолютно семантично різних зображень, вносячи деякі зміни до зображень-запитів. Людське сприйняття може навіть не побачити, що зображення було навмисно видозмінено, і буде виглядати як звичайне.

2.5 Системи глибокого хешування

Сімейство алгоритмів, які використовують глибоке хешування, досить велике, оскільки існує декілька підходів до пошуку схожих зображень за допомогою хеш-кодів. Розглянемо найбільш відомі, та цікаві з практичної точки зору.

2.5.1 Глибоке попарне хешування

Лі та ін. [24] пропонують новий метод глибокого хешування під назвою Deep Pairwise-Supervised Hashing (DPSH), який одночасно тренує мережу та функцію хеш-коду з використанням деяких парних зображень з мітками за допомогою використанням зворотного зв’язку. Мережа тренується на парах зображень, які мають мітки, котрі визначають подібність зображень. Як і в стандартному глибокому хешуванні першим кроком є вилучення ознак зображень, потім йде шар хеш-функції, і потім функція втрат, яка оцінює мітки вхідної пари зображень, після чого мережа робить висновок як їй змінити ваги. Тобто частина вилучення ознак зображень і хеш-функція це єдина наскрізна мережа, яка тренується як будь-яка звичайна мережа. Ідея такої мережі тісно перекликається з Сіамськими нейронними мережами.

2.5.2 Глибоке хешування на основі триплетів

На основі роботи Лі та інших, було запропоновано модифікацію, яку Вонг та інші [20] назвали глибоким хешуванням на основі триплетів (англ. Triplet-based Deep Hashing TBDH). Цей метод використовують таку трійку даних - два схожих зображення і одне несхоже. Таким чином ваги моделі підлаштовуються таким чином, щоб схожі зображення отримали якомога схожі хеші, а от третє зображення мусить отримати зовсім інший хеш.

2.5.3 Класична мережа глибокого хешування

Мережа глибокого навчання Deep Hashing Network (DHN) - це деяке поєднання згаданих мереж в цій роботі, але з однією головною відмінністю - ця мережа не тільки генерує хеш-код для заданих зображень, а ще видає передбачення стосовно того чи є схожими ці зображення. Таку конфігурацію використовують частіше ніж DPSH та TBDH.

2.5.4 Хешування на основі прихованого фактору

Вже згадувався метод LSH, який використовувався для задачі ANN, і визначався як окремий алгоритм. Тепер розглянемо альтернативу, яка називається хешування на основі прихованого фактору Latent Factor Hashing (LFH) [15].

На відміну від LSH, хешування на основі прихованого фактору вбудовується як окремий шар в нейрону мережу, що допомагає зробити одну цілісну мережу, без розбиття на підзадачі. Ці алгоритми дуже схожі між собою, але розбиття на сегменти відбувається випадковим чином лише на ініціалізації мережі, а далі мережа під час навчання корегує згенеровані гіперплощини. LFH є передостаннім шаром в мережах DPSH, TBDH, DHN.

2.5.5 Сіамська нейронна мережа

Сіамська нейронна мережа (англ. Siamese Neural Network (SNN)) - це особлива архітектура нейронних мереж, яка приймає на вхід одразу два елемента, і робить передбачення на основі цих елементів разом. Частіше за все такі мережі роблять порівняння наскільки схожі два елементи, тому використовуються для пошуку подібності або задач пошуку найближчого сусіда. Прикладами застосування таких мереж є розпізнавання обличчя, пошук схожого підпису серед представлених, або просто порівняння схожості зображень. Звичайно пропускати попарно зображення-запит з кожним із бази даних не потрібно, оскільки оброблені зображення вже містять попередньо обчислений вектор ознак. Використовуючи такий метод, як LSH, можна легко знайти схожі зображення.

Архітектура такої нейронної мережі складається із двох однакових згорткових нейронних мереж (наприклад, VGG16, ResNet50 тощо) відповідно для кожного вхідного елемента/зображення. Однак ваги для цих мереж спільні, тому вихідні вектори ознак будуть схожими (наприклад, за косинусною мірою), що дозволяє дізнатися ступінь подібності.

Розділ 3. РЕАЛІЗАЦІЯ АЛГОРИТМУ

3.1 Характеристика алгоритму

Алгоритм спирається на принципах глибокого хешування, щоб знаходити схожі зображення зразки, по яким відбувається визначення класів. Алгоритм в ідеальних умовах може фактично продовжувати онлайн навчання (адаптивна модель), однак власне корегування вагів моделі не відбувається, тому таке навчання можна назвати «псевдоадаптивним». Головною перевагою мережі є те, що вона може розширювати кількість класів і випадків, і це може бути реалізовано через зворотній зв'язок від реальних користувачів. Також мережа виконує вимогу конфіденційності зображень користувачів, оскільки не використовує зображення напряму, а дістає з них ознаки і перетворює їх на хеш-код, який може передаватися безпосередньо на сервер.

Загальне застосування алгоритму може бути таким:

1) мережа завантажується на девайс користувача;
2) зображення-запит проходить через мережу;
3) отриманий хеш-код відправляється на сервер;
4) за допомогою хеш-коду і проіндексованій базі хеш-код – класи, визначаються класи конфіденційності вмісту.

Для того щоб продовжити навчання на основі зворотнього зв’язку від користувачів, знадобиться разом з хеш-кодом зображення передавати й класи зображення (це можуть бути і нові класи, яких нема в на сервері). Новий хеш-код додається до хеш-таблиць, і наступного разу коли він стане найближчим сусідом до зображення-запиту, його клас вплине на результат.

Див. Додаток Д, щоб ознайомитися з результатами псевдоадаптивності.

3.2 Набір даних

В даній роботі процес навчання нейронної мережі проходить на основі “Набору даних візуальної конфіденційності” (англ. The Visual Privacy (VISPR)

Dataset). Цей набір даних включає 68 різних класів, які у свою чергу групуються у 11 основних категорій. Найбільш цікавими класами є “номерний знак”, “кредитна карта”, “паспорт”, “водійське посвідчення”, “студентський квиток”, “підпис”, “повна голота”, “релігія” тощо. З повним переліком усіх класів і категорій можна ознайомитись у Додатку А. Даний набір даних використовувався у досліджені Трибхуванеш О. та інших [12]. Варто зазначити, що класи не є взаємовиключними, тому більшість фотографій відносяться до декількох класів одночасно. Найбільший клас «safe» (3711 зображень), тобто ті зображення, які є безпечними і не містять конфіденційної інформації. Загальна кількість зображень становить 10 000. Джерелом цих зображень є OpenImages датасет, Flickr (фотохостинг), а також Twitter. Всі зображення мають відкриту ліцензію на використання і розповсюдження. Незважаючи, на велику кількість зображень, 37 класів мають менше ніж 40 зразків, що робить даний набір незбалансованим.

Див. Додаток Б, щоб побачити приклади з набору даних.

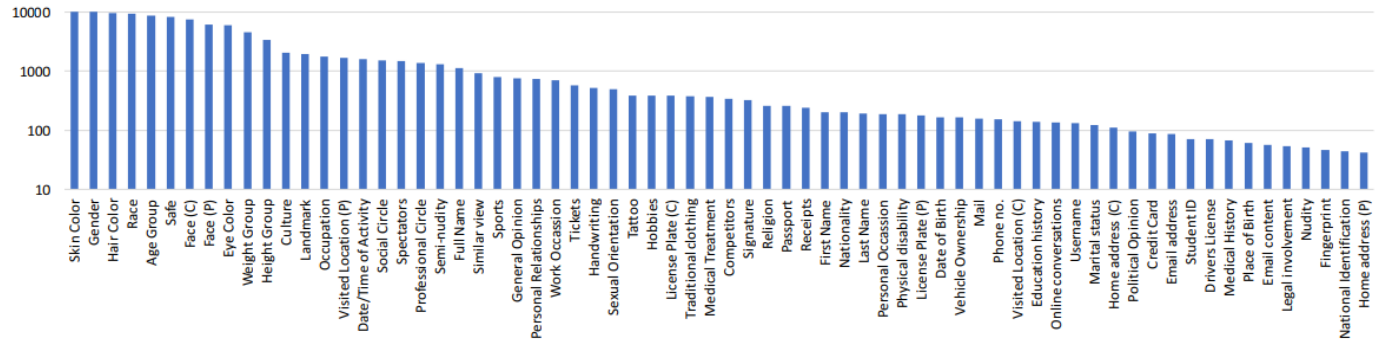


Рисунок 3.1 – Розподіл кількості зображень по категоріям [12].

3.3 Опис запропонованого алгоритму

Запропонований алгоритм об'єднує у собі декілька методів та концепцій, які розглядаються у Розділі 1 та 2. Перелік етапів алгоритму наступний:

1) трансферне навчання;
2) вилучення ознак;
3) хешування ознак (LSH);
4) пошук найближчих сусідів;

5) визначення ймовірності класів.

3.3.1 Етап 1: Трансферне навчання

Першим кроком є вибір претренованої нейронної мережі, яку будемо використовувати для вилучення ознак. Наразі існує дуже багато готових архітектур, однак не всі вони підходять для всіх задач. Мережі з великою кількістю шарів можуть давати кращий результат, однак витрачають на це більше обчислювальних ресурсів і вимагають більше часу на повний прохід даних. Ще важливим критерієм є простір, який мережа займає на диску, бо деякі мережі можуть займати по пів гігабайту, що звичайно не є дружнім для кінцевого користувача, якому прийдеться дуже довго чекати поки застосунок завантажиться.

В роботі розглядалося 3 претреновані мережі - ResNet34, ResNet50, VGG16. VGG16 була відкинута через великий розмір - 528 МБ. ResNet34 більш старий варіант архітектури ніж ResNet50, тому остаточно було обрано мережу ResNet50. Розмір ResNet50 становить 98 МБ, що є більш менш припустимим розміром. Претреновані мережі можна отримати з бібліотеки fastai.

Сам процес трансферного навчання можна розбити на такі кроки:

1) підготовка набору даних;
2) завантаження претренованої мережі;
3) заміна останніх повнозв'язних шарів на власні в кінці мережі;
4) тренування лише повнозв'язних шарів, залишаючи всі інші шари замороженими;
5) тренування всієї мережі, розморожуючи всі шари.

Щоб підібрати найкращий темп навчання, використаємо метод описаний в Розділі 1. Для кроку 4 було визначено темп навчання 0.002, та кількість епох - 4. Для кроку 5 було визначено темп навчання 0.000025, і кількість епох - 4.

Варто зауважити, що процес додаткового навчання на визначеному наборі даних є опціональним кроком. При бажанні можна використати претреновану мережу відразу, і результати також будуть задовільними. Однак, цей крок допомагає

підлаштувати мережу, щоб вона краще розуміла власне визначений набір даних - зображення з одного класу будуть наближатися і скупчуватися один до одного у N-вимірному просторі ознак, а з різних класів - відштовхуватись один від одного. Без додаткового навчання, мережа все рівно буде знаходити схожі зображення, однак навчальний набір на якому тренували претреновану мережу має бути досить вичерпним і узагальненим. Наприклад, якщо в претренованому наборі не було зображень паспортів, то мережі буде складно пов'язувати такі схожі зображення в групи.

3.3.2 Етап 2: Вилучення ознак

Після того, як нейронна мережа натренована, ми можемо вилучити ознаки для кожного зображення із набору даних. Ці ознаки - це представлення зображення як N-вимірний вектор. Таким чином ми можемо представити зображення в N-вимірному просторі, а тому чим ближче дане зображення до іншого зображення, тим вірогідніше, що ці зображення схожі. Цей вектор можна вилучити з передостаннього повнозв'язного шару. Щоб спростити процес вилучення, використовується так званий Pytorch Hooks, що дозволяє перехоплювати значення конкретного шару без додаткового коду.

Передостанній повнозв’язний шар мережі ResNet50 генерує вектор розмірністю 512.

3.3.3 Етап 3: Хешування ознак (LSH)

Щоб підготувати згенеровані ознаки для задачі пошуку найближчих сусідів, наступним кроком буде хешування ознак за допомогою алгоритму LSH. Без цього кроку знадобилось би попарне порівняння зображення-запиту з кожним зображенням з набору даних, що звичайно є обчислювально дорогою операцією. Для хешування використаємо бібліотеку lshashpy3, яка містить реалізацію LSHash. LSHash - це клас, який відповідає за хешування та індексацію елементів. В даному класі вхідними параметрами є k - розмір хеш-коду, L - кількість хеш-таблиць, та d - розмірність вектору ознак. Для задачі були визначені такі значення (k, L, d) - $(10, 5, 512)$.

3.3.4 Етап 4: Пошук найближчих сусідів

Після формування LSHash, можна вже шукати найближчих сусідів для зображення-запиту. Для цього використаємо вбудовану функцію в LSHash - query. Функція повертає усіх найближчих сусідів для даного зображення. Кількість повернутих зображень залежить від параметрів (k, L, d). Наприклад, якщо k велике, то ймовірність колізій менша, а тому для запиту може повернутися мало або нуль схожих зображень, і навпаки, при великих значеннях - може повернутися майже весь набір даних. Схожа ситуація для параметру L: при великій кількості таблиць, алгоритм буде повертати малу кількість запитів, оскільки перетин результатів всіх хеш-таблиць буде давати малу або порожню множину.

3.3.5 Етап 5: Визначення ймовірності класів

Маючи найближчих сусідів та їх класи можна визначити класи для деякого запиту, та ймовірності приналежності до цих класів. Припустимо ми маємо множину найближчих сусідів NN, де кожен елемент $x_i = (img_i, c_i)$, img_i - зображення з обробленого набору зображень, c_i - множина класів, яка належить до цього зображення з визначеного набору класів C. Щоб визначити класи для зображення-запиту q, визначимо ймовірність приналежності до кожного класу, як відношення кількісті зображень з заданим класом до загальної кількості сусідів. Див Додаток В і Г щоб ознайомитися з результатами.

3.4 Можливі недоліки алгоритму

Недоліком, який може заважати адаптивному навчанню, є випадок коли новий клас перетинається з уже наявними зображеннями у базі. Якщо, припустимо, до цього у нас були 1000 зображень котів, і всі вони не визначалися як один клас, то при спробі додати зображення-зразок, який представляє клас «кіт», це не сильно вплине на майбутні передбачення, оскільки інші зображення котів будуть заважати точному визначенню цього класу.

Також, дуже важко передбачити усі класи і випадки, і щоб зібрати таку вичерпну базу зображень треба мати дуже велику базу користувачів та багато часу. При цьому, дуже важко контролювати користувачів, оскільки на нейрону мережу можуть бути здійснені атаки. Також користувачі можуть навмисно додавати небезпечні зображення, як безпечні, що вплине на майбутні передбачення мережі.

3.5 Бібліотека fast.ai

Fastai — це сучасна бібліотека для машинного навчання, яка є у вільному доступі на GitHub за ліцензією Apache 2. Бібліотеку можна встановити за допомогою conda або pip (менеджерів пакетів для python).

Бібліотека має велику спільноту, яка піклується про вичерпну документацію, практичні приклади, навчальні ресурси. Бібліотеці присвячено багато книжок, але найбільш відома є книжка засновника бібліотеки [8].

Головні переваги fastai, які вирізняють цю бібліотеку серед інших, це доступність і легконалаштовуваність. Інші бібліотеки, як правило, змушують робити вибір між лаконічністю та гнучкістю, що робить їх або складними для розуміння, або важкими для програмування. Творці бібліотеки fastai намагалися поєднати швидкість розробки, як у Keras, і можливість налаштування, як у PyTorch. Поєднати особливості двох популярних бібліотек мотивувала авторів на створення гарної архітектури і дизайну бібліотеки. В результаті було отримано високорівневу архітектуру, тобто користувачеві не потрібно заглиблюватись на низький рівень, щоб кастомізувати бібліотеку під свої потреби.

Високий рівень абстракції стає у нагоді для початківців і спеціалістів суміжних спеціальностей, які хочуть якнайшвидше отримати результати, щоб застосувати їх на практиці. Бібліотека включає у себе модулі для обробки зображень, текстів, табличних даних та часових рядів.

Головним класом в бібліотеці є Learner (учень), який представляє собою композицію із всіх інших класів, необхідних для навчання та прогнозування. У Learner передається архітектура нейронної мережі - це може бути або вже готова

претренована мережа, або нова мережа, створена користувачем. У Learner включається і оптимізатор (наприклад Adam), а також передається DataLoader - клас, який відповідає за загрузку і представлення навчальних даних. Інтеграція всіх задач в один клас дозволяє fastai задавати каркас для більш гнучких систем, і водночас допомагає легко розібратися на початку знайомства з бібліотекою. Це допомагає початківцям, і таким чином вони не роблять помилок, і мають повне представлення що за чим слідує. Крім того, у fastai вбудовано багато допоміжних функцій, які допомагають візуалізувати дані, і допомагають більш якісно розібратися та проаналізувати підготовлений набір даних. Також, fastai передбачає можливість використання різних методів для покращення машинного навчання, такі як пакетна нормалізація (англ. batch-normalization) [13], заморозка шарів, дискримінаційні показники навчання (англ. discriminative learning rates) [10] тощо. Загалом, використання інтегрованих за замовчуванням методів передбачає меншу кількість коду.

Бібліотека побудована на основі PyTorch (Paszke et al. 2017), NumPy (Oliphant, n.d.), PIL (Clark and Contributors, n.d.), Pandas (McKinney 2010) та різних інших бібліотек. Оскільки головним ядром бібліотеки залишається PyTorch, у бібліотеці можна безпосередньо працювати з основними типами та класами PyTorch.

Висновки

Проблема збереження персональної конфіденційності користувачів наразі є на слуху, і більшість компаній намагаються впровадити міри безпеки, щоб не дозволити поширенню конфіденційної інформації з боку наївних користувачів. Запропонована робота піднімає питання конфіденційності зображень, і намагається вирішити задану проблему.

У ході виконаної роботи було створено систему, яка дозволяє вирізняти та класифікувати конфіденційні зображення. Запропонований алгоритм має властивість «псевдоадаптивності», тобто можливість продовжувати збільшувати кількість класів і випадків для класів, не змінюючи ваги нейронної мережи. Також система зберігає конфіденційність зображень, оскільки працює з хеш-кодами, а не з зображеннями напряму.

Для використання та розробки системи використовувались вже готові високорівневі рішення, я саме бібліотека Fast.ai та PyTorch, і як середовище використовувався сервіс Google Colabs.

Було досліджено поточний стан алгоритмів машинного та глибокого навчання, способи покращення та види навчання. Розглянуті випадки, коли набір даних є замалим, що обмежує використання традиційного підходу машинного навчання, та варіанти як, використовуючи такі набори, натренувати досить потужну мережу (N-кадрове навчання). Проаналізовано задачу пошуку найближчих сусідів, і варіант покращення даної задачі за допомогою хешування (LSH). Було визначено предмет і проблему дослідження, і власне визначення конфіденційності персональних зображень. Також, згадано про атаки на нейронні мережі, їх різновиди та суть, і деякі способи захисту від них.

Дослідження включає у собі такі поширені задачі, як багатокласову класифікацію (MLC), наближений пошук найближчих сусідів (APN), пошук подібних зображень, колізії під час хешування, вилучення ознак зображення, атаки на нейроні мережі.

Наступним кроком для покращення дослідження може слугувати тема федеративного навчання, що може допомогти покращити результати за рахунок створення незалежної нейронної мережі на основі реальних зображень користувачів для конкретної предметної області (наприклад для соціальних мереж, або для месенджерів).

Список літератури

1. Cakebread C. People will take 1.2 trillion digital photos this year – thanks to smartphones [Електронний ресурс] / Caroline Cakebread // Business Insider. – Режим доступу: https://www.businessinsider.com/12-trillion-photos-to-be-taken-in-2017-thanks-to-smartphones-chart-2017-8 (дата звернення: 27.06.2022). – Назва з екрана.

2. Company [Електронний ресурс] // Pinterest Newsroom. – Режим доступу: https://newsroom.pinterest.com/en/company (дата звернення: 27.06.2022). – Назва з екрана.

3. Content-based image retrieval: a review of recent trends [Електронний ресурс] // Taylor & Francis. – Режим доступу: https://www.tandfonline.com/doi/full/10.1080/23311916.2021.1927469 (дата звернення: 27.06.2022). – Назва з екрана.

4. Deep learning [Електронний ресурс] // Deep Learning. – Режим доступу: http://www.deeplearningbook.org (дата звернення: 27.06.2022). – Назва з екрана.

5. Dey S. Fast and accurate personal identification based on iris biometric [Електронний ресурс] / Somnath Dey, Debasis Samanta // International journal of biometrics. – 2010. – Т. 2, № 3. – С. 250. – Режим доступу: https://doi.org/10.1504/ijbm.2010.033389 (дата звернення: 27.06.2022). – Назва з екрана.

6. Emilio Soria Olivas. Handbook of research on machine learning applications and trends: algorithms, methods and techniques / Emilio Soria Olivas ; ред.: Jose David Martin Guerrero [та ін.]. – [Б. м. : б. в.], 2009.

7. Generalizing from a few examples: a survey on few-shot learning [Електронний ресурс] / YAQING WANG [та ін.]. – [Б. м. : б. в.], 2020. – (Препринт ; arXiv:1904.05046v3). – Режим доступу: https://arxiv.org/pdf/1904.05046.pdf (дата звернення: 27.06.2022). – Назва з екрана.

8. Howard J. Deep learning for coders with fastai and pytorch: AI applications without a phd / Jeremy Howard, Sylvain Gugger. – [Б. м.] : O'Reilly Media, Incorporated, 2020. – 624 с.

9. Ian Goodfellow. Deep learning (adaptive computation and machine learning series) / Ian Goodfellow, Yoshua Bengio, Aaron Courville. – [Б. м. : б. в.], 2016.

10. Jeremy Howard. Universal language model fine-tuning for text classification [Електронний ресурс] / Jeremy Howard, Sebastian Ruder. – [Б. м. : б. в.], 2018. – (Препринт ; arXiv:1801.06146v5). – Режим доступу: https://arxiv.org/pdf/1801.06146.pdf (дата звернення: 27.06.2022). – Назва з екрана.

11. Lee K. M. Locality-Sensitive hashing techniques for nearest neighbor search [Електронний ресурс] / Keon Myung Lee // International journal of fuzzy logic and intelligent systems. – 2012. – Т. 12, № 4. – С. 300–307. – Режим доступу: https://doi.org/10.5391/ijfis.2012.12.4.300 (дата звернення: 27.06.2022). – Назва з екрана.

12. Orekondy T. Towards a visual privacy advisor: understanding and predicting privacy risks in images [Електронний ресурс] / Tribhuvanesh Orekondy, Bernt Schiele, Mario Fritz // 2017 IEEE international conference on computer vision (ICCV), Venice, 22–29 жовт. 2017 р. – [Б. м.], 2017. – Режим доступу: https://doi.org/10.1109/iccv.2017.398 (дата звернення: 27.06.2022). – Назва з екрана.

13. Sergey Ioffe. Batch normalization: accelerating deep network training by reducing internal covariate shift [Електронний ресурс] / Sergey Ioffe, Christian Szegedy. – [Б. м. : б. в.], 2015. – (Препринт ; arXiv:1502.03167). – Режим доступу: https://arxiv.org/pdf/1502.03167.pdf (дата звернення: 27.06.2022). – Назва з екрана.

14. Smith L. N. Cyclical learning rates for training neural networks [Електронний ресурс] / Leslie N. Smith // 2017 IEEE winter conference on applications of computer vision (WACV), Santa Rosa, CA, USA, 24–31 берез. 2017 р. – [Б. м.], 2017. – Режим доступу: https://doi.org/10.1109/wacv.2017.58 (дата звернення: 27.06.2022). – Назва з екрана.

15. Supervised hashing with latent factor models [Електронний ресурс] / Peichao Zhang [та ін.] // SIGIR '14: the 37th international ACM SIGIR conference on research and development in information retrieval, Gold Coast Queensland Australia. – New York, NY, USA, 2014. – Режим доступу:

https://doi.org/10.1145/2600428.2609600 (дата звернення: 27.06.2022). – Назва з екрана.

16. Surmenok P. Estimating an optimal learning rate for a deep neural network [Електронний ресурс] / Pavel Surmenok // Medium. – Режим доступу: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0 (дата звернення: 27.06.2022). – Назва з екрана.

17. The perceptron - jonty sinai [Електронний ресурс] // Jonty Sinai. – Режим доступу: https://jontysinai.github.io/jekyll/update/2017/11/11/the-perceptron.html (дата звернення: 27.06.2022). – Назва з екрана.

18. Thomas A. C. What is artificial neural networks (anns)? [Електронний ресурс] / Arun C. Thomas // Medium. – Режим доступу: https://medium.com/the-ultimate-engineer/artificial-neural-networks-anns-for-the-rest-of-us-172c2aa96127 (дата звернення: 27.06.2022). – Назва з екрана.

19. Towards effective deep embedding for zero-shot learning [Електронний ресурс] / Lei Zhang [та ін.] // IEEE transactions on circuits and systems for video technology. – 2020. – Т. 30, № 9. – С. 2843–2852. – Режим доступу: https://doi.org/10.1109/tcsvt.2020.2984666 (дата звернення: 27.06.2022). – Назва з екрана.

20. Wang X. Deep supervised hashing with triplet labels [Електронний ресурс] / Xiaofang Wang, Yi Shi, Kris M. Kitani // Computer vision – ACCV 2016. – Cham, 2017. – С. 70–84. – Режим доступу: https://doi.org/10.1007/978-3-319-54181-5_5 (дата звернення: 27.06.2022). – Назва з екрана.

21. What are convolutional neural networks? [Електронний ресурс] // IBM - Deutschland | IBM. – Режим доступу: https://www.ibm.com/cloud/learn/convolutional-neural-networks (дата звернення: 27.06.2022). – Назва з екрана.

22. What are neural networks? [Електронний ресурс] // IBM - Deutschland | IBM. – Режим доступу: https://www.ibm.com/cloud/learn/neural-networks (дата звернення: 27.06.2022). – Назва з екрана.

23. What is the optimizer Adadelta? [Електронний ресурс] // Peltarion. – Режим доступу: https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimizers/adadelta (дата звернення: 27.06.2022). – Назва з екрана.

24. Wu-Jun Li. Feature learning based deep supervised hashing with pairwise labels [Електронний ресурс] / Wu-Jun Li, Sheng Wang, Wang-Cheng Kang. – [Б. м. : б. в.], 2016. – (Препринт / Department of Computer Science and Technology, Nanjing University ; arXiv:1511.03855v2). – Режим доступу: https://arxiv.org/pdf/1511.03855.pdf (дата звернення: 27.06.2022). – Назва з екрана.

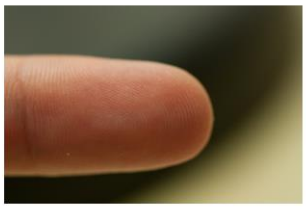
Додаток А

(обов’язковий)

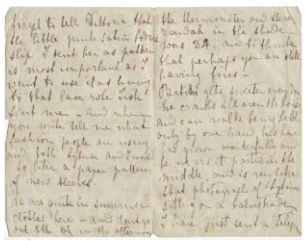
Перелік категорій і класів

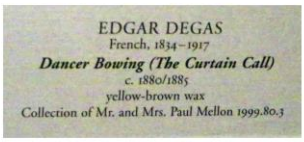
Безпечні		
Safe	Не містить конфіденційної інформації.	
Персональна інформація		
Gender	Стать суб’єкта чітко видно.	
Age	Можна вирізнити вікову категорію суб’єкта.	

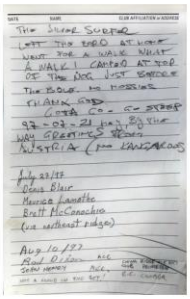
Weight	Можна вирізнити вагову категорію суб’єкта	
Height	Можна визначити зріст суб’єкта.	
Eye Color	Якщо очі видно та можна визначити їх колір: коричневий, горіховий, блакитний або зелений.	
Hair Color	Видно колір волосся на голові суб’єкта.	

Fingerprint	Відбиток пальця видно на знімку крупним планом.	
Signature	Повний підпис видно на зображенні.	
Face (Complete)	Повністю видно обличчя. Також включає фотографії обличчя на посвідченнях особи, документах або рекламних щитах.	
Face (Partial)	Видно менше 70% обличчя або є оклюзія, наприклад, коли об’єкт носить сонцезахисні окуляри.	

Tattoo	Суб'єкт має татуювання.	
Nudity (Partial)	Суб'єкт в нижній білизні.	
Nudity (Complete)	Суб’єкт повністю оголений.	
Race	Можна вирізнити расу до якої належить суб’єкт.	

(Skin) Color	Колір шкіри людини можна відрізнити.	
Traditional Clothing	Суб’єкт з’являється в одязі, який вказує на певний регіон чи країну, наприклад. дирндль, сарі.	
Full Name	Повне ім’я, яке з’являється в контексті бланка чи документа. Також включає, якщо ім'я можна визначити з підпису.	
Name (First)	Тільки якщо ім’я видно на бланку, документі, значку чи одязі.	

Name (Last)	Тільки якщо прізвище видно на бланку, документі, значку чи одязі.	
Place of Birth	Місце народження вказано чітко, наприклад, у формі або в документі, що посвідчує особу.	
Date of Birth	Дата народження чітко вказана в письмовій формі. Включає рік, місяць або день народження.	
Nationality	Чітко видно паспорт із зазначенням країни. Включає випадок, якщо суб’єкт з прапором країни або в уніформі з прапором (наприклад, солдат або міжнародний спортсмен).	

Handwriting	Рукописний текст на будь-якій поверхні.	
Marital status	Суб'єкт носить обручку. Включає весільні фотографії нареченого і нареченої.	
Документи		
National Identification	Такі документи, як зелена карта або європейське національне посвідчення особи, за винятком паспортів.	
Credit Card	Лицьова або зворотна сторона кредитної картки. Включає випадки, коли картку частково видно, напр. в чиїсь руці або в подрібненому вигляді	

Passport	Фотографія будь-якої сторінки в паспорті або його лицьовій обкладинці.	
Drivers License	Лицьова або зворотна сторона або збоку посвідчення водія.	
Student ID	Лицьова або зворотна сторона студентського посвідчення особи, яка має чітко читатися принаймні назву школи, коледжу чи університету.	
Mail	Вміст листа або конверта.	

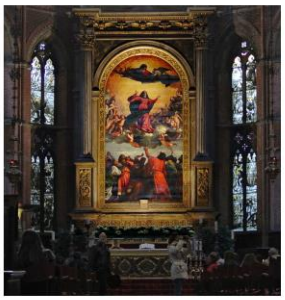
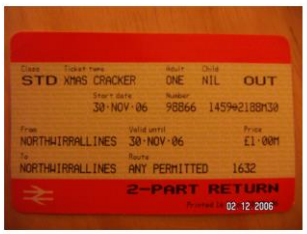
Receipts	Квитанції про покупку, що вказують на фінансову операцію з чітко видимою сумою, напр. ресторанний чек.	
Tickets	Квиток на проїзд, кіно чи концерт, який визначає місце подорожі чи подію.	
Здоров’я		
Physical disability	Суб'єкт з постійною фізичною інвалідністю, наприклад. особа з ампутацією або людина в інвалідному візку.	
Medical Treatment	Суб’єкт або з травмою, або з госпіталізацію.	

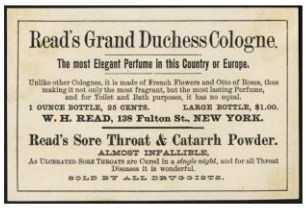
Medical History	Фотографії ліків або медичних рецептів.	
Працевлаштування		
Occupation	Суб’єкт в уніформі, що вирізняє певну професію, напр. лікар, поліцейські, будівельник.	
Work Occasion	Суб'єкт фотографується під час виступу, презентації, відвідування пов'язаної з роботою або трансляції події. Включає фотографії людей у офіційному вбранні в офісі.	
Особистий життя		
Religion	Суб'єкт з'являється із помітним релігійним символом, одягом, що відповідає релігії, або в релігійному місці	

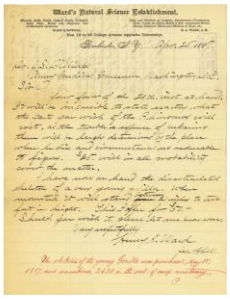
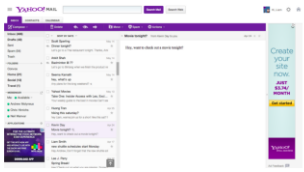
Sexual Orientation	Два об’єкти фотографуються в інтимній обстановці	
Culture	Суб’єкти з’являються, коли святкують традиційний фестиваль або відвідують діяльність, пов’язану з мистецтвом чи культурою, наприклад. концерт, грати.	
Hobbies	Видно непрофесійну діяльність суб'єкта напр. гра на музичному інструменті, фотографування.	
Sports	Суб’єкт бере участь у спортивних заходах у приміщенні чи на свіжому повітрі.	

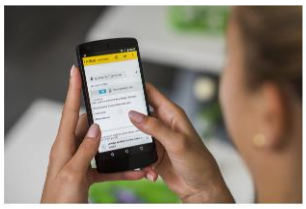
Education history	Фотографії містять вказівки на історію освіти суб'єкта, наприклад, на церемонії випуску, одяг, що вказує на університет або академічний чи шкільний атестат	
Legal involvement	Фотографії, що вказують на причетність суб'єкта до діяльності, пов'язаної із законом, наприклад когось заарештовують у судовому засіданні.	
Personal Occasion	Фотографії людей, які святкують особисту подію з друзями або членами сім'ї, наприклад весілля, день народження.	
General Opinion	З'являється тема, пов'язана з плакатом або одягом, що вказує думку щодо загальних питань, наприклад. війни, податки, права меншинств.	

Political Opinion	Суб'єкт з плакатом або в натовпі на політичній акції.	
Відносини		
Personal Relationships	Фотографії людей у візуально ідентифікованих особистих стосунках, напр. мати-син, чоловік-дружина.	
Social Circle	Суб’єкти сфотографовані в невимушеній обстановці, наприклад друзі на вечірці, разом гуляють по вулиці.	
Professional Circle	Група людей, які мають спільну роботу (наприклад, група поліцейських) або одягнені для професійного заходу (наприклад, конференції чи зустрічі).	

Competitors	Група людей, які беруть участь у командних видах спорту. Також включає випадок, коли суб'єкти належать до однієї команди.	
Spectators	Група людей, які спостерігають за подією, як-от концерт або виставу	
Similar view	Група людей на мітингу чи акції протесту, які поділяють думки щодо загального питання. Включає лише випадок, коли видно плакати чи одяг, що позначаю справу чи мітинг для політичної партії.	
Місцезнаходження		
Visited Landmark	Фотографія містить текст, що вказує назву компанії, вуличний знак або добре відомий орієнтир.	

Visited Location (Complete)	Текст із зазначенням повної адреси (наприклад, квитанція ресторану з адресою ресторану) або знімок екрана з місцезнаходженням на основі GPS.	
Visited Location (Partial)	Текст, який частково вказує місце розташування об’єкта, наприклад назва вулиці, місто чи країна, де була зроблена фотографія.	
Home address (Complete)	Фотографія, що містить повну некомерційну поштову адресу.	
Home address (Partial)	Фотографія, що містить часткову некомерційну поштову адресу.	

Date/Time of Activity	Фотографія містить інформацію про дату та/або час місцезнаходження або активності об’єкта, наприклад водяний знак з відміткою часу на зображенні або годинник на фотографії.	
Phone no.	Номер телефону, який видно на фотографії (особистий або комерційний).	
Інтернет-діяльність		
Username	Знімок екрана веб-сайту, на якому згадується будь-яке ім’я користувача чи Інтернет-дескриптори.	
Email address	Будь-яка повна дійсна адреса електронної пошти, яка відображається на фотографії або знімку екрана.	

Email content	Знімки екрана електронних листів, включаючи тему листа або частини вмісту листа.	
Online conversations	Скріншоти онлайн-розмов, дописів, твітів або активності в Інтернеті будь-якого користувача	
Автомобіль		
Vehicle Ownership	Фотографія людини, яка їздить на автомобілі.	
License Plate (Complete)	Добре помітний номерний знак або реєстраційний номер будь-який автомобільний транспортний засіб.	

License Plate (Partial)	Частковий номерний знак або реєстраційний номер будь-якого транспортного засобу	

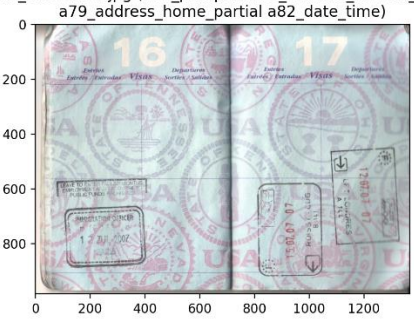
Додаток Б

(обов’язковий)

Приклад класів

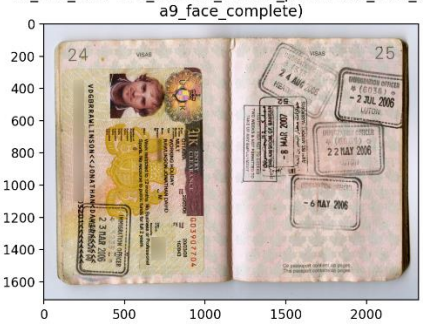
2017_63282699.jpg (a16_race a17_color a1_age_approx a2_weight_approx a3_height_approx a4_gender a55_religion a6_hair_color a9_face_complete)

2017_57338699.jpg (a31_passport a75_address_current_partial a79_address_home_partial a82_date_time)



2017_63848224.jpg (a17_color a25_nationality a31_passport)

2017_87269852.jpg (a16_race a17_color a19_name_full a1_age_approx a31_passport a4_gender a5_eye_color a6_hair_color a75_address_current_partial a82_date_time a9_face_complete)



2017_57186913.jpg (a16_race a17_color a19_name_full a1_age_approx a23_birth_city a24_birth_date a25_nationality a31_passport a3_height_approx a4_gender a75_address_current_partial a82_date_time a8_signature a9_face_complete)

2017_57216313.jpg (a103_license_plate_complete)

2017_26770838.jpg (a103_license_plate_complete)

2017_12900460.jpg (a25_nationality a31_passport)

2017_17830961.jpg (a19_name_full a25_nationality a26_handwriting a31_passport a46_occupation a75_address_current_partial a82_date_time a8_signature)

2017_74882865.jpg (a31_passport a75_address_current_partial a82_date_time a8_signature)

2017_31295075.jpg (a17_color a31_passport a75_address_current_partial a82_date_time)

2017_24347642.jpg (a10_face_partial a16_race a17_color a1_age_approx a2_weight_approx a3_height_approx a46_occupation a4_gender a55_religion a5_eye_color a64_rel_personal a6_hair_color a9_face_complete)

Додаток В

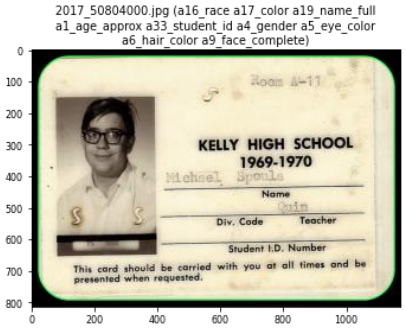
(обов’язковий)

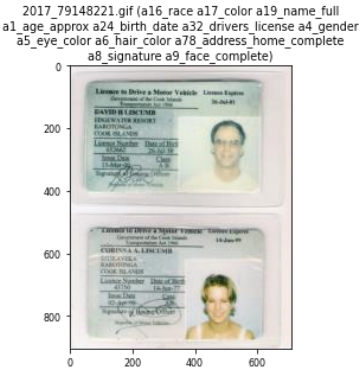
Приклад 1: пошук схожих зображень і визначення категорії

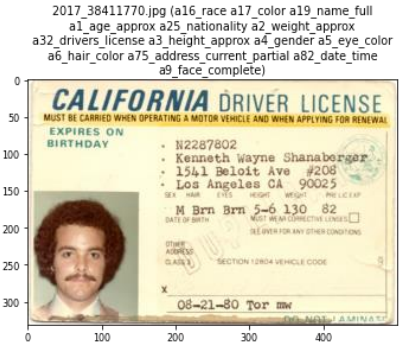
Визначені категорії (перше зображення): {

```
'a31_passport': 0.21030554931923604,
'a17_color': 0.08471059884175304,
'a16_race': 0.08418987092148861,
'a4_gender': 0.08372395646651513,
'a9_face_complete': 0.08353562302166917,
'a6_hair_color': 0.08328799820768909,
'a5_eye_color': 0.08124477709739589,
'a19_name_full': 0.07811227433887073,
'a1_age_approx': 0.07738618715932799,
'a32_drivers_license': 0.04784261438318936,
'a24_birth_date': 0.03822862639096512,
'a8_signature': 0.03725496552237413
```

}







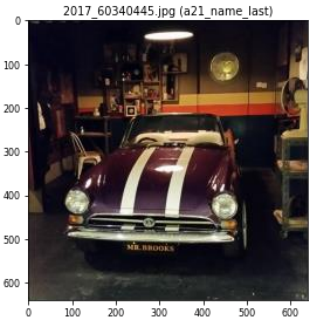
Додаток Г

(обов’язковий)

Приклад 2: пошук схожих зображень і визначення категорії

Визначені категорії (перше зображення): {

```
    'a104_license_plate_partial': 0.60765153006115125,
    'a103_license_plate_complete': 0.3904818228091337
}
```



Додаток Д
(обов’язковий)
Приклад псевдоадаптивності

Визначені категорії (перше зображення): {

```
    'a0_safe': 0.6915838799726031,
    'cat': 0.214128645851671
}
```

Пояснення: клас «cat» було додано, як зворотній зв’язок – додано лише одну фотографію, однак цього достатньо щоб для схожої фотографії (перше зображення) було визначено клас «cat».