КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теорії та технології програмування

Кваліфікаційна робота
на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки
на тему:

ВЕЛИКОМАСШТАБНИЙ ПОШУК СХОЖИХ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ГЛИБОКОГО МЕТРИЧНОГО НАВЧАННЯ

Виконала студентка 2-го курсу магістратури
Андрейчук Анастасія Віталіївна

(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Панченко Тарас Володимирович

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«____» _____ 2021 р.,
протокол № ___
Завідувач кафедри
М. С. Нікітченко _____

(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 63 сторінки, 37 ілюстрацій, 5 таблиць, 36 джерел посилань.

ГЛИБОКА НЕЙРОННА МЕРЕЖА, ГЛОБАЛЬНІ ОЗНАКИ, КОМП'ЮТЕРНИЙ ЗІР, ЛОКАЛЬНІ ОЗНАКИ, МЕТРИЧНЕ НАВЧАННЯ, ПОШУК СХОЖИХ ЗОБРАЖЕНЬ, РОЗПОДІЛЕНЕ НАВЧАННЯ.

Об'єктом роботи є великомасштабна задача пошуку зображень, схожих на задане, за допомогою глибокого метричного навчання. Предметом роботи є створена система на основі глибокої нейронної мережі для знаходження зображень, схожих на задане, у масивній базі даних.

Метою роботи є розробка системи на основі глибокої нейронної мережі і її навчання для знаходження зображень, схожих на задане, у масивній базі даних.

Методи розробки: навчання глибокої нейронної мережі, семантична та геометрична верифікація, розподілене навчання, планування значень параметрів, алгоритми комп'ютерного зору. Інструменти розробки: Python v3.7, TensorFlow v2.4.1, OpenCV v4.5.2, Numpy v1.17.1, SciPy v1.6.2, RAPIDS cuml v0.19.0.

Результати роботи: запропоновано покращення методу тренування нейронної мережі для збільшення виразності та дискримінативності глобальних та локальних ознак, виділених із зображення, в порівнянні з останніми сучасними моделями; покращення швидкості навчання та утилізації обчислювальних ресурсів; покращення етапу ранжування результатів за допомогою локальних ознак, виділених із зображення, використовуючи семантичну верифікацію, та покращення гіперпараметрів геометричної верифікації.

Розроблена мережа може використовуватися для пошуку схожих зображень у, наприклад, пошукових системах чи галереях зображень, коли користувач не може визначити об'єкт пошуку в текстовому вигляді, а також для визначення поточної локації роботами.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ЗСТ	Загальна середня точність
GLD2	Google Landmark Dataset v2
TPU	Тензорний процесор, Tensor Processing Unit
FLOPS	Кількість операцій з рухомою комою на секунду, floating point operations per second

ВСТУП

Проблема пошуку схожих зображень – це пошук семантично узгоджених або подібних зображень, схожих на задане, у базі зображень, за допомогою аналізу їх візуального змісту. Ця робота розв’язує великомасштабну задачу пошуку схожих зображень за допомогою глибокого метричного навчання. Під час виконання роботи було розроблено глибоку нейронну мережу, а також натреновано її на конкретному наборі даних, що містить більше чотирьох мільйонів зображень.

Оцінка сучасного стану об’єкта розробки. Означену проблему почали розглядати ще на початку тисячоліття у спільноті комп’ютерного зору, і за останні двадцять років було досягнуто значного прогресу. Усі запропоновані підходи можна розділити на два періоди: інженерія ознак та вивчення ознак (ознакою вважається певна особливість, що притаманна об’єкту на певному зображенні та не притаманна іншим об’єктам).

Яскравим прикладом першого підходу є методи, що базуються на SIFT [1] та SURF [2]. Такі системи пошуку схожих зображень порівнюють локальні ознаки зображення з усіма зображеннями в базі за допомогою таких технік, як Bag-of-Words [3], використовуючи візуальні слова, отримані кластеризацією локальних ознак та оцінкою TF-IDF, а потім роблять геометричну верифікацію за допомогою RANSAC [4].

Другий підхід, що є більш актуальним, розпочинається з 2012 року з введення глибоких згорткових нейронних мереж на прикладі AlexNet [5]. Це було проривним кроком у використанні нейронних мереж для складних задач комп’ютерного зору, адже глибокі мережі можуть вивчити ознаки напряму з даних через декілька рівнів абстракції. Основна ідея підходу вивчення ознак полягає в тому, що тренується глибока нейронна мережа, яка бере зображення на вхід та видає вектор ознак, що бувають локальними і глобальними, таким чином, щоб схожі зображення мали схожі ознаки. Так, наприклад, [6] використовували глобальні ознаки, отримані за допомогою NetVLAD [7] для

локалізації кімнат та аудиторії. На сьогодні моделі DELG [8] та DELF [9] показують найкращі результати у розв’язанні поставленої задачі.

Актуальність роботи та підстави для її виконання. У сучасному світі з експонеційно зростаючим обсягом зображень та відеофайлів розробка систем, що ефективно оперують ними, є дуже важливою. Однією з задач таких систем (галерей, пошукових систем (рис. 1) тощо) є можливість пошуку зображень за заданим, адже потенціал такої задачі є нескінченним – як-от застосування для ідентифікації особи, визначення місцезнаходження, пошук медичних зображень тощо. Пошук потрібного зображення зазвичай вимагає пошуку серед великої кількості зображень (тисяч або мільйонів), тому ефективність пошуку є не менш важливим фактором, ніж точність.

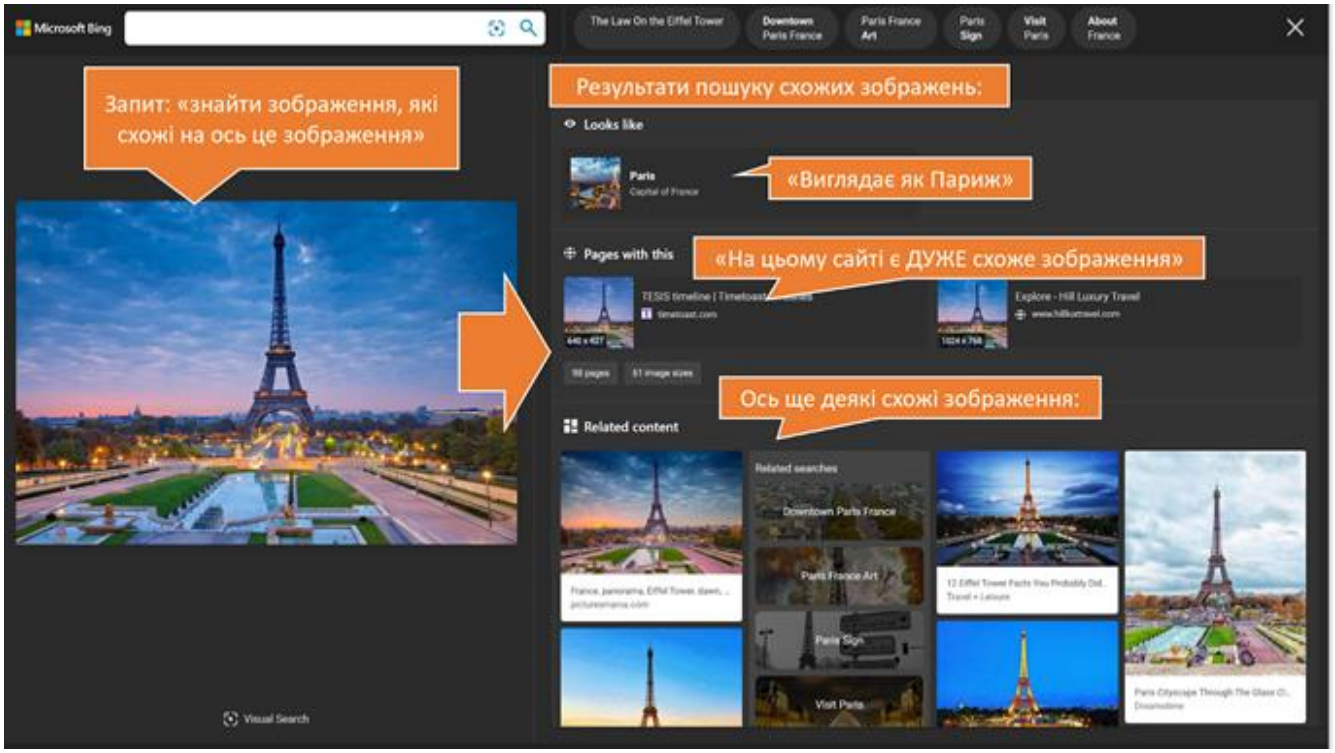

Рисунок 1 – Приклад застосування таких методів: пошукова система Бінг

Мета й завдання роботи. Метою роботи є розробка системи на основі глибокої нейронної мережі та її навчання для знаходження зображень, схожих на задане, у масивній базі даних. В таку систему входить:

- Система семантичної верифікації для фільтрування вхідного запиту (зображення, до якого шукаються схожі).
- Глибока нейронна мережа, що використовується для обчислення глобальних та локальних ознак зображень.
- Система локальної (геометричної) верифікації для ранжування отриманих результатів (тобто схожих зображень у базі даних).

Об’єкт та предмет роботи, методи й засоби розробки. Об’єктом роботи є великомасштабна задача пошуку зображень, схожих на задане, за допомогою глибокого метричного навчання. Предметом роботи є створена система на основі глибокої нейронної мережі для знаходження зображень, схожих на задане, у масивній базі даних.

Методи розробки: навчання глибокої нейронної мережі, семантична та геометрична верифікація, розподілене навчання, планування значень параметрів. Засоби розробки: Python v3.7, TensorFlow v2.4.1, OpenCV v4.5.2, Numpy v1.17.1, SciPy v1.6.2, RAPIDS cuml v0.19.0.

Можливі сфери застосування. Розроблена мережа може використовуватися для пошуку схожих зображень у, наприклад, пошукових системах чи галереях зображень, коли користувач не може визначити об’єкт пошуку в текстовому вигляді, а також для визначення поточного місцезнаходження роботами.

РОЗДІЛ 1 ЗАДАЧА ПОШУКУ СХОЖИХ ЗОБРАЖЕНЬ

1.1. Локальні та глобальні ознаки

Для точного та ефективного пошуку в масивному наборі зображень основою є виділення компактних, але описових ознак. Будемо розглядати кожне із зображень через його глобальні та локальні ознаки (дескриптори). Це дає вищу точність у пошуку зображень, адже ці ознаки фокусуються на різних деталях зображення. Дескриптор кодує зображення особливим чином, даючи змогу йому бути порівняним та відповідати іншим зображенням.

Локальний дескриптор описує ділянку зображення, виділяючи патерни, що відрізняються від околиць. Для порівняння зображень можна проаналізувати попарні відповідності між дескрипторами, проте для великомасштабного порівняння це не є ефективним. Також не всі отримані ознаки є дискримінаційними, і неглибокі класифікатори допомагають вибрати з них підходящі. Зокрема, геометрична інформація про конкретні області зображень зберігається саме в локальних дескрипторах, і вони є доволі точними для твердих об’єктів на зображенні.

Зазвичай кілька локальних дескрипторів використовується для задачі порівняння зображення, і це є більш надійним, оскільки не всі дескриптори повинні збігатися для порівняння. Це забезпечує стійкість до змін між відповідними зображеннями. Локальні ознаки інваріантні до масштабування зображення, його обертання або зміни освітлення.

Локальні дескрипторі мають бути:

1. Повторювані та точні. Таким чином, вони можуть бути отримані з різних зображень одного й того самого об’єкта.
2. Характерні для зображеного об’єкта, тобто інші об’єкти не будуть їх мати.

Незважаючи на те, що аналіз локальних дескрипторів ознак дає змогу отримати єдину векторну репрезентацію зображення, це можна також здійснювати

безпосередньо за допомогою глобальних дескрипторів, тобто таких, що кодують властивості зображення в цілому, а не ділянок. Оскільки вони обробляють повне зображення, то глобальні дескриптори не вимагають фази виявлення ознак, що робить обчислення швидшим. Таким чином, зміст зображення можна компактно узагальнити за допомогою глобального дескриптора, не фокусуючись на просторовому розташуванні об’єктів на світлині.

Порівняно з поданням у локальних дескрипторах, глобальні дескриптори менш стійкі до шуму, але вони дають перевагу в чутливості, тоді як локальні – в точності.

1.2. Набір даних для навчання

Задача пошуку схожих зображень не обмежує область застосування, адже вхідне зображення для пошуку по ньому може бути будь-яким. У цій роботі розроблена мережа для демонстрації її роботи та порівняння результатів з іншими публікаціями навчається на наборі даних, що містить локації, і повертає результати пошуку для зображень, що входять до цього обсягу, і фільтрує інші.

1.2.1. Google Landmark Dataset v2

Google Landmark Dataset v2 (GLD2), представлений в [10], на поточний момент є останнім еталонним набором даних для задачі великомасштабного пошуку та розпізнавання зображень, охоплюючи визначні пам’ятки зі всього світу. Він є найбільш складним для навчання і масивним серед наборів, що використовуються для метричного навчання.

Набір GLD2 складається з трьох піднаборів:

1. Тренувальний набір для задачі розпізнавання зображень, що містить 4.1 мільйони зображень 203 тисяч визначних пам’яток.

2. Тренувальний набір для задачі пошуку зображень, що містить 762 тисячі зображень 101 тисячі визначних пам’яток.

3. Тестовий набір, що містить 118 тисяч зображень (з правильними відповідями для обох задач).

Для імітації реальної ситуації, коли зображення для пошуку задається користувачем (наприклад у візуальних пошукових системах чи в програмах для альбомів фотографій), у тестовому наборі всього один відсоток зображень є визначними пам’ятками. Таким чином, перевірка моделі на цьому тестовому наборі дає не тільки точність класифікації, а й надійність (мала кількість хибно позитивних результатів, коли вхідне зображення не є визначною пам’яткою, але модель вважає її такою). Іншим фактором, що наближає GLD2 до ситуацій у реальному світі, є різноманітність зображень серед кожного класу. Наприклад, фотографії можуть бути зроблені всередині чи зовні визначної пам’ятки (якщо це будівля тощо), або навіть містити краєвид, що видно при знаходженні на цій пам’ятці безпосередньо (фізично). Також, звичайно, на світлинах об’єкт розташований із різних боків та на різній відстані, а також при різному освітленні. Для деяких пам’яток до їх класу відносять світлини, що тільки опосередковано пов’язані з пам’ятками, такі як, наприклад, фотографії архітекторів або картини в музеї.

Набір даних був побудований відповідно до факту, що світлин більш відомих визначних пам’яток значно більше порівняно з менш відомими, і цей розподіл зберігається і в GLD2, роблячи класи сильно незбалансованими. Таким чином, набір даних має велику кількість не дуже відомих локальних пам’яток: максимум 5 фотографій доступно для 38 % класів, а 10 – для 57 % (рис. 2).

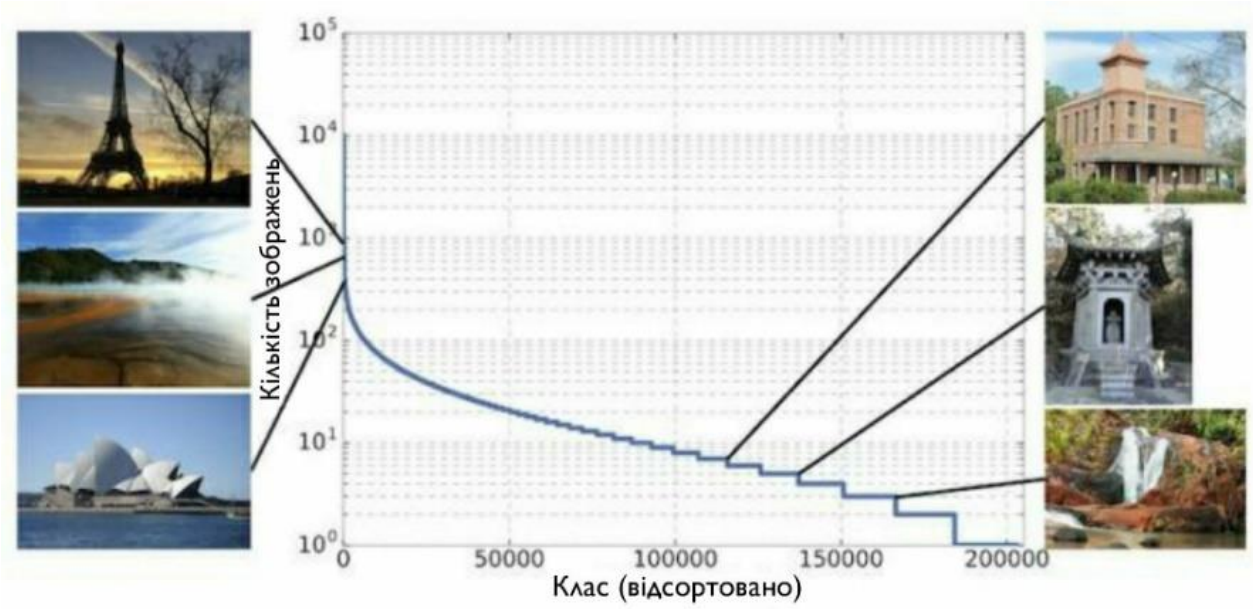

Рисунок 2 – Розподіл класів по відомості

1.3. Етапи пошуку схожих зображень

Глобальні та локальні ознаки, що можна виділити із зображень, потрібно аналізувати окремо, адже вони описують різні характеристики зображення. Глобальні ознаки розглядаються насамперед, адже, по-перше, вони описують зображення в більш широкому сенсі, а по-друге їх набагато ефективніше порівнювати в масивній базі, оскільки глобальний дескриптор перетворює зображення у вектор, а локальний – у набір векторів з їх координатами. А за допомогою локальних ознак на набагато меншому наборі можна вже побачити, які із зображень найбільше схожі на задане в ключових точках.

Таким чином, процес безпосередньо пошуку схожих зображень виконується в три етапи (рис. 3):

1. Означене зображення (запит) представляється у вигляді глобальних ознак, тобто векторного подання його вмісту.
2. У базі даних шукаються вектори зображень, що є найближчими до даного в певній метриці.
3. Під час післяобробки результати сортуються (ранжуються) за їх релевантністю (використовуючи локальні ознаки).

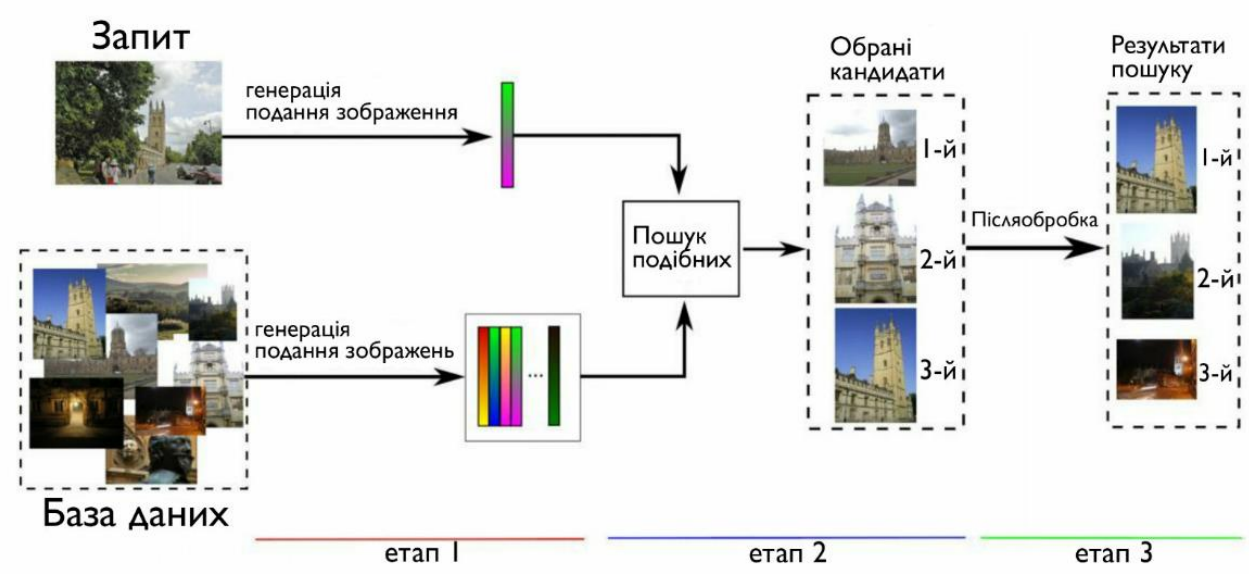

Рисунок 3 – Етапи пошуку зображень, схожих на задане

РОЗДІЛ 2 ВИДІЛЕННЯ ГЛОБАЛЬНИХ ОЗНАК

2.1. Глибоке метричне навчання

Одними з найбільш ранніх алгоритмів, що все ще використовуються в галузі машинного навчання, є навчання на основі подібності, що мають на основі ідею здатності знаходити подібності між різними об’єктами.

Заради можливості виміру подібності між заданими екземплярами треба мати відстань для встановлення міри, що показує, чи одна пара об’єктів є більш схожою між собою, ніж інша пара. Незважаючи на існування нескінченної кількості відстаней, не всі вони підходять для використання в цій задачі, і вибір відстані, що може коректно адаптуватися до даних, є найважливішим для означеного типу алгоритмів.

Так само як ціль звичайного машинного навчання – при заданому наборі вхідних даних та відповідних їм класів виявити набір правил або деяку складну функцію, що виводить із вхідного даного коректний клас, ціллю метричного навчання є вивчення функції подібності із вхідних даних. Вивчення векторів ознак у такому вигляді означає, що відстань від вхідних даних, що належать одному класу, зменшується, а між тими, що належать до різних, збільшується, що є головною метою метричного навчання.

Метричне навчання – це підхід, що базується на дистанційній метриці, що намагається встановити подібність чи розбіжність між вхідними даними. Глибоке метричне навчання, у свою чергу, використовує нейронні мережі для того, щоб вивчити дискримінаційні ознаки із вхідних даних і застосування метрики до них.

Глибоке метричне навчання (рис. 4) набагато краще встановлює нелінійні залежності за допомогою вивчення нелінійних перетворень у просторі ознак, порівняно з метричним навчанням, і в останні роки було показано [11], що воно дає гарні результати для багатьох задач комп’ютерного зору, зокрема у випадках, коли надається зображення одного й того самого об’єкта з різних сторін, при різному освітленні та в інших умовах.

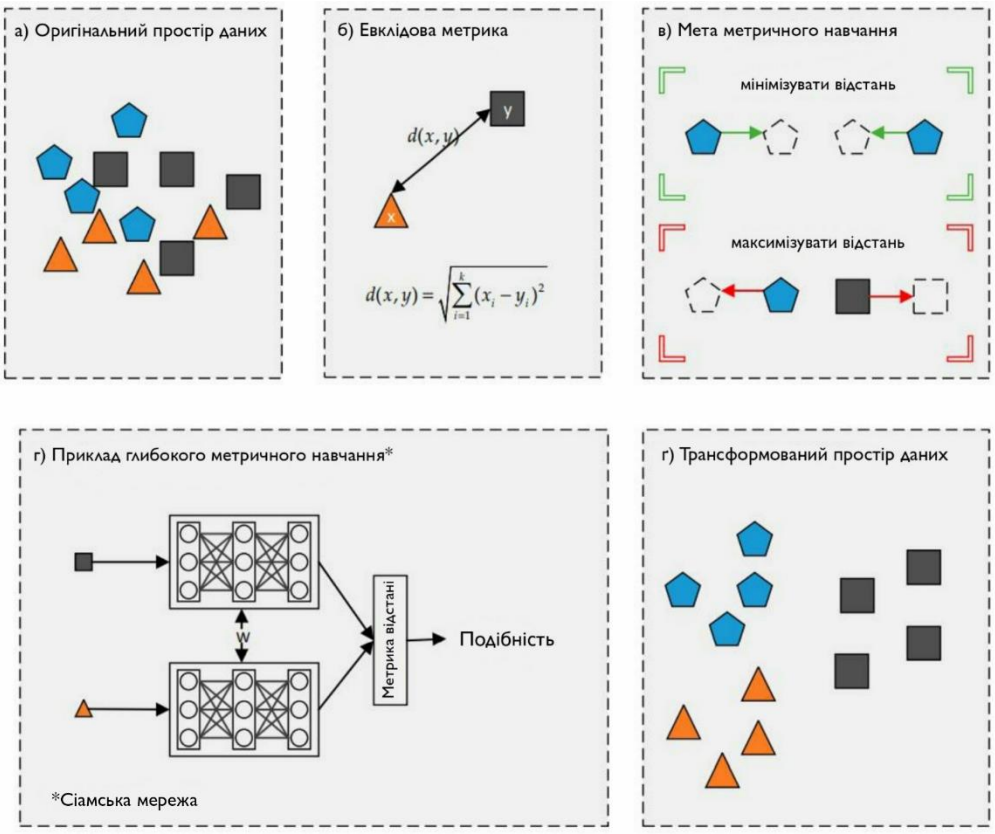

Рисунок 4 – Глибоке метричне навчання

2.2. Глибокі залишкові нейронні мережі (ResNet)

Глибокі залишкові нейронні мережі (ResNet) були представлені в [12] та є подібними до згорткових нейронних мереж, крім наявності тотожного зв’язку між шарами. Зазвичай залишкові мережі містять дво- або тришарові блоки з нелінійностями (зазвичай ReLU) та нормалізацією пакетів. На рис. 5 видно тотожний зв’язок як стрілку справа, що йде від початку й до кінця залишкового блоку.

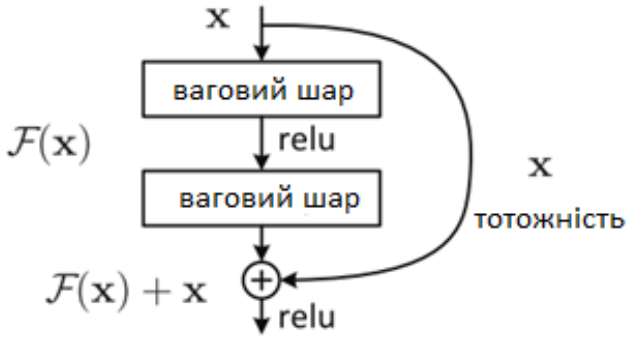

Рисунок 5 – Залишковий блок глибоких залишкових мереж

Залишкове відображення – це різниця між входом і виходом залишкового блоку, можна сказати, що це кількість помилок, яку можна додати до входу для досягнення наближення кінцевої функції (рис. 6).

Рисунок 6 – Інтуїтивне зображення залишкового відображення

Оскільки глибокі залишкові мережі розроблялися для вирішення проблеми зникнення градієнта, саме для цього й потрібна наявність залишкового блоку. Послідовні шари не призводять до погіршення продуктивності мережі, адже при доданні тотожних зв’язків на будь-яку мережу функціональність архітектури не зміниться. Таким чином, помилка глибокої мережі не може бути більшою за помилки її неглибоких частин. Таким чином, замість наближення $H(x)$ послідовними шарами наближується залишкове відображення $\mathcal{F}(x) = H(x) - x$.

Існує два типи тотожних зв’язків:

1. Тотожні зв'язки (**X**) можна використовувати безпосередньо, якщо вхідні та вихідні дані мають однакові розміри:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

2. При зміні розмірів:

a) зв’язок все одно виконує відображення тотожності, де при збільшенні розмірності додаються нульові елементи (заповнення нулями);

b) проекційний зв’язок використовується для узгодження розмірності (виконується згортка 1x1) за такою формулою:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$

У першому випадку не потрібні додаткові параметри, а в другому треба W_s. Через це складність обчислень для глибоких залишкових мереж майже однакова зі складністю для простих глибоких мереж.

2.2.1. Друга версія ResNet

У другій версії [13] автори оригінальної публікації [12] вдосконалили залишковий блок із використанням перед-активації вагових шарів замість після-активації (рис. 7).

Тут друга нелінійність виконує роль тотожного відображення, для того щоб результат від додавання між тотожним відображенням та залишковим відображенням можна було передати до наступного блоку для подальшої обробки. У той час у першій версії до суми спочатку застосовується нелінійність, і потім отриманий результат передається в подальший блок.

Залишковий блок другої версії можна використовувати для будь-якої з архітектур глибоких залишкових мереж, таких як, наприклад, ResNet 152, утворюючи ResNet 152 V2.

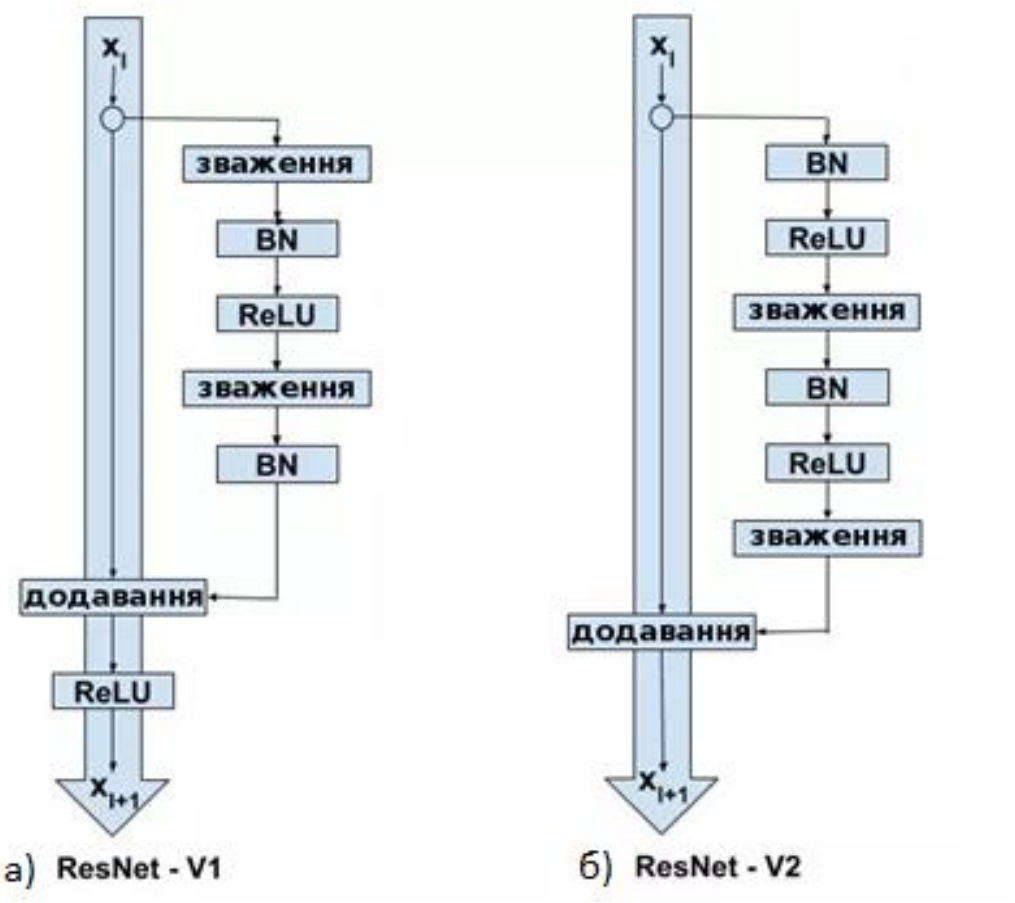

Рисунок 7 – Порівняння схем: а –ResNet версія 1; б – ResNet версія 2

2.2.2. ResNet 152

Архітектура ResNet 152 має декілька етапів, як показано на схемі нижче. На вхід мережі подається зображення розмірності $W \times H$, де W, H кратні 32, та з шириною 3 (кількістю каналів). В основі лежить дизайн із використанням вузьких місць. Для кожної залишкової функції $\mathcal{F}$ три шари (згортки 1×1, 3×3, 1×1) складаються послідовно. З них перший та третій відповідають за зменшення та відновлення розмірів відповідно, а другий – є вузьким місцем з меншою розмірністю вводу та виводу. У кінці мережа має шар середньої агрегації, після якої слідує повнозв’язний шар з тією кількістю нейронів, що потрібен для виходу задачі, що розглядається. Цю архітектуру візуалізовано на рис. 8.

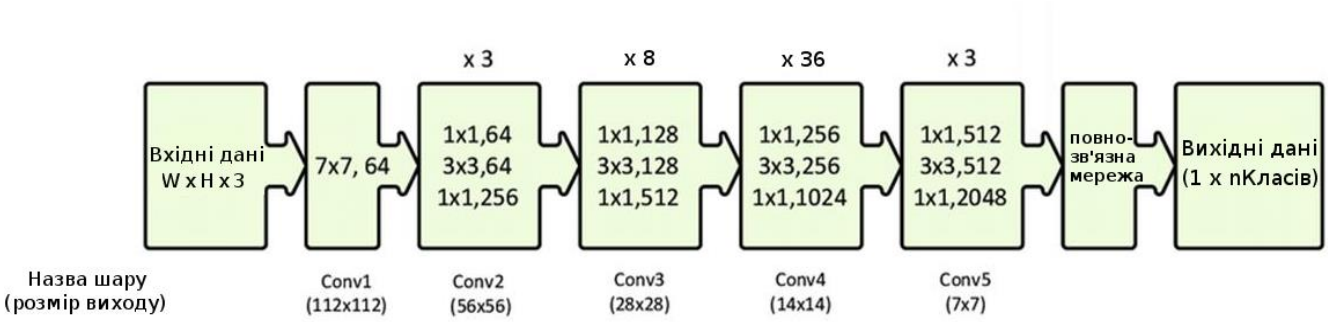

Рисунок 8 – ResNet 152

2.3. EfficientNet

EfficientNet, запронована в [14], є однією з найновіших архітектур нейронних мереж. Вона показує найвищу на даний момент точність на наборі даних ImageNet [15], і при цьому є у декілька разів менше та швидше інших точних нейронних мереж, таких як ResNet 152.

2.3.1. Змішане масштабування (Compound Scaling)

Відмінність цієї архітектури від інших полягає в тому, що зазвичай для поліпшення точності в моделях збільшують кількість каналів (ширина), шарів (довжина) або покращують роздільну здатність вхідного зображення. Хоч це і допомагає, урешті-решт, ефективність мережі знижується через завелику кількість параметрів.

При одночасному збалансованому збільшенні всіх вищенаведених параметрів із фіксованим коефіцієнтом, як це зроблено в EfficientNet, уся мережа масштабується більш рівномірно. Це інтуїтивно зрозуміло через той факт, що при більшому розмірі вхідного зображення (його роздільної здатності) необхідно працювати з ним, маючи більшу кількість шарів та каналів для можливості врахувати всі патерни на цьому зображенні. Такий механізм має назву змішаного масштабування (рис. 9).

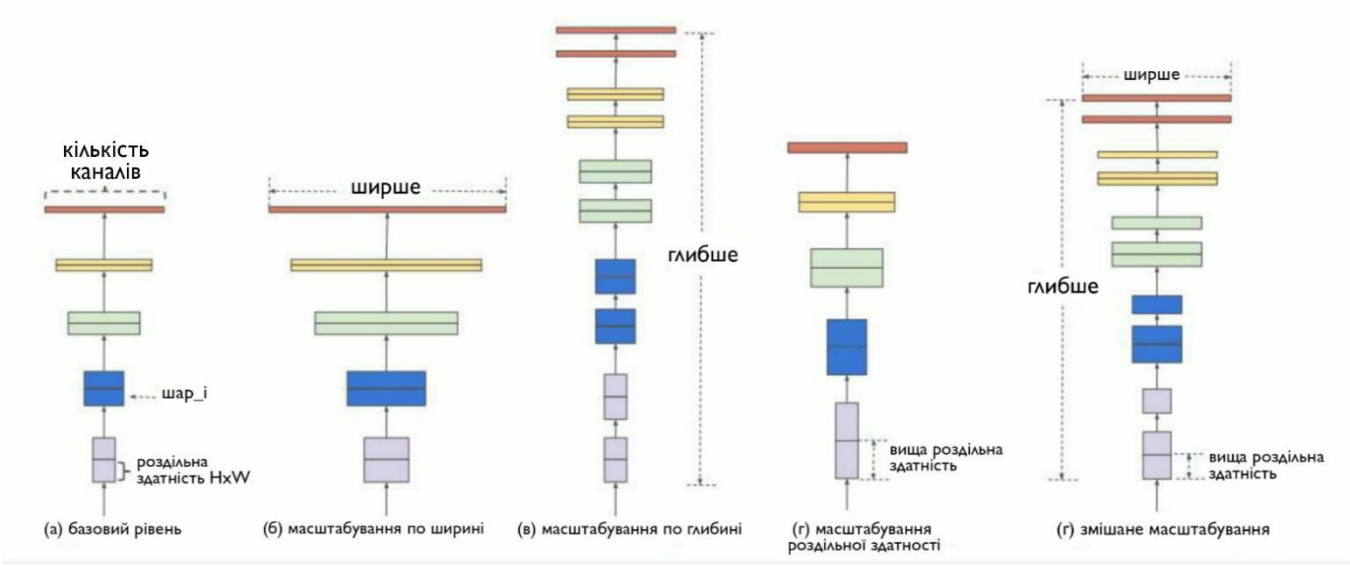

Рисунок 9 – Масштабування моделі: а – початкова мережа; б–г – збільшення тільки одного з параметрів: ширини, довжини та роздільної здатності відповідно; ґ – змішане масштабування.

2.3.2. Базова модель EfficientNet-B0

Незважаючи на те, що змішане масштабування можна застосувати до вже існуючих архітектур нейронних мереж, таких як ResNet, автори [14] запропонували свою модель. EfficientNet насправді є групою архітектур нейронних мереж, що має базову мережу, від якої походять всі інші за допомогою змішаного масштабування. Такою базовою моделлю є EfficientNet-B0, архітектура якої була отримана за допомогою підходу під назвою Пошук Нейронної Архітектури (Neural Architecture Search), що був запропонований у [16].

Суть цього підходу полягає в тому, що багатокритеріальна оптимізація та підкріплене навчання використовуються для пошуку архітектури нейронної мережі, що відповідає параметрам оптимізації, в цьому випадку це точність та кількість операцій з рухомою комою на секунду (FLOPS). Модель EfficientNet-B0 була отримана при максимізації такої функції:

$$ACC(m) \times [FLOPS(m)/T]^w,$$

де $T = 400\,000\,000$ – цільова FLOPS, m – модель, $w = -0.07$ – гіперпараметр для балансування точності та FLOPS.

Архітектура EfficientNet-B0 описана в табл. 1:

Таблиця 1 – Базова мережа EfficientNet-B0

Етап i	Оператор $\hat{\mathcal{F}}_i$	Роздільна здатність $\hat{H}_i \times \hat{W}_i$	Кількість каналів $\hat{C}_i$	Кількість шарів $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Для спрощення візуалізації представимо її в іншому вигляді (рис. 13), увівши поняття кореня мережі (рис. 10), заключних шарів (рис. 10), підблоків (рис. 11) та модулів (рис. 12).

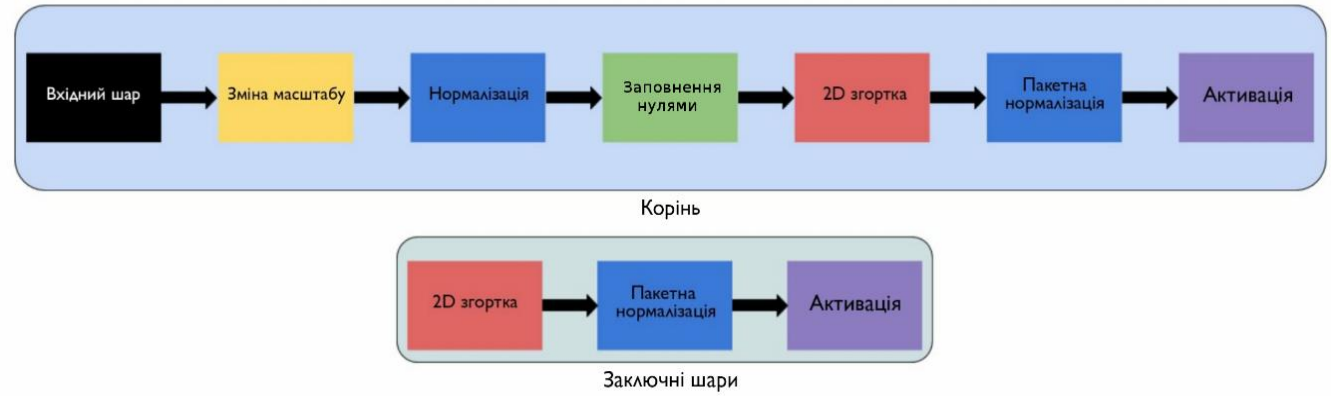

Рисунок 10 – Корінь та заключні шари

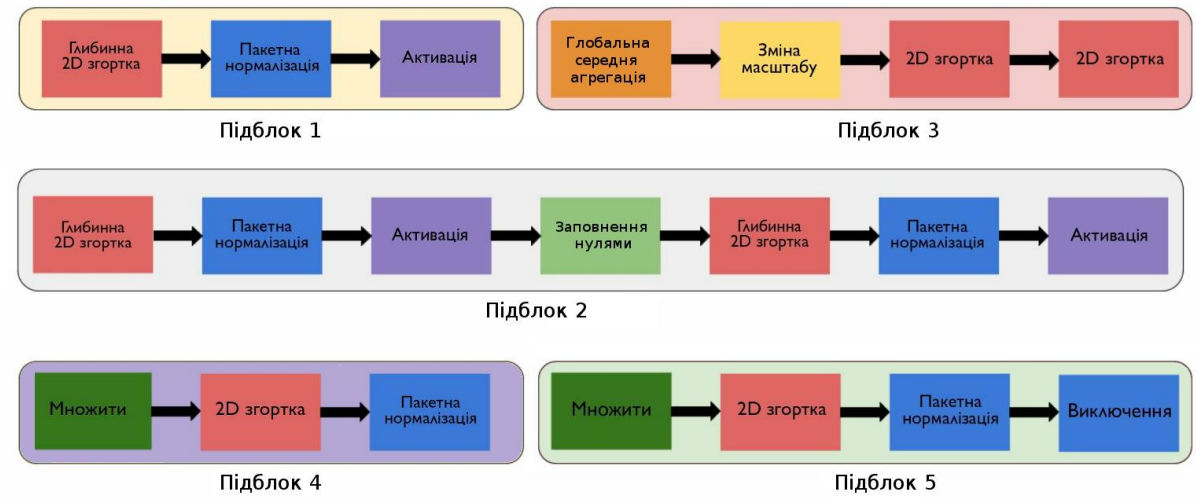

Рисунок 11 – Підблоки

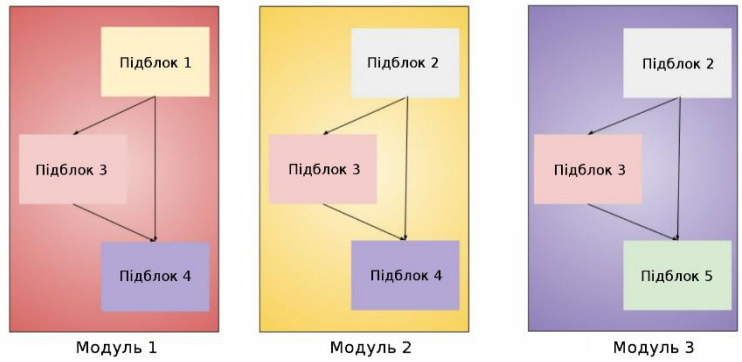

Рисунок 12 – Модулі

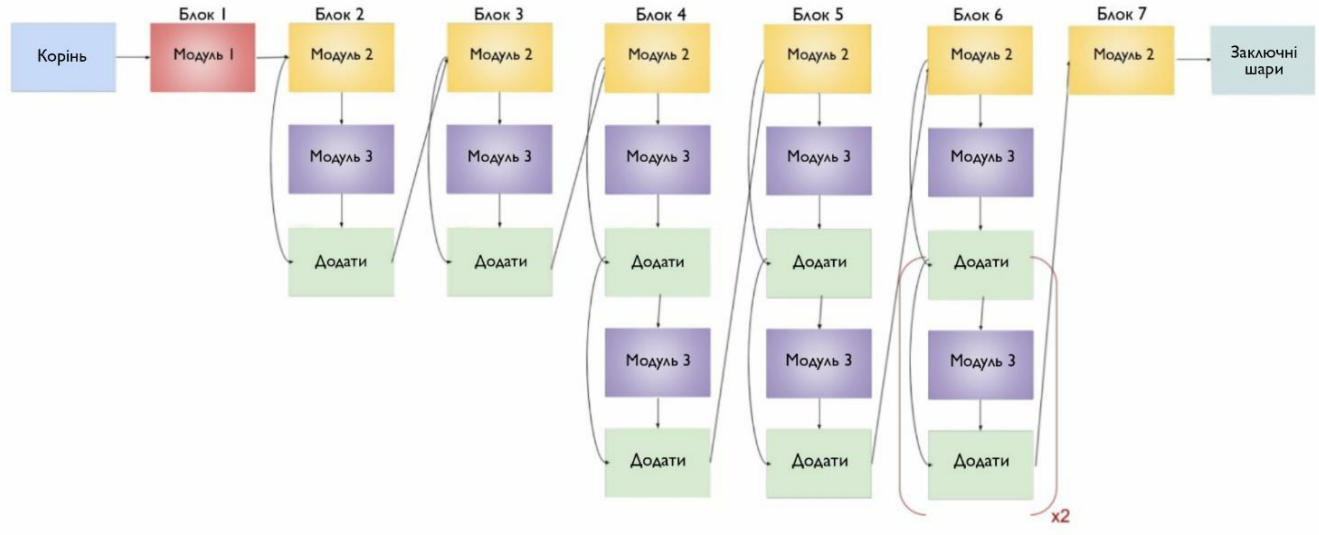

Рисунок 13 – EfficientNet-B0

2.3.3. Масштабування EfficientNet-B0 до B1-B7

Нехай гіперпараметри d – глибина мережі, ω – ширина мережі, r – роздільна здатність зображення, причому виконується наступне:

$$d = \alpha^{\varphi}, \omega = \beta^{\varphi}, r = \gamma^{\varphi}, \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$(\alpha \geq 1, \beta \geq 1, \gamma \geq 1)$$

Можна сказати, що φ – це коефіцієнт, що задається користувачем та визначає, скільки є доступних додаткових ресурсів. Константи $\alpha,\ \beta,\ \gamma$ визначають відповідний розподіл цих ресурсів по гіперпараметрах, і їх можна знайти за допомогою малого пошуку по ґратці. Таким чином, можна масштабувати гіперпараметри базової мережі для отримання більшої. Це все виконується в два кроки:

1. Вважаючи, що вдвічі більше ресурсів доступно, було зафіксовано $\varphi = 1$ та знайдено значення $\alpha = 1.2,\ \beta = 1.1,\ \gamma = 1.15$ при обмеженні $\alpha * \beta 2 * \gamma 2 \approx 2$.
2. Для отримання EfficientNet-B1 до B7 при зафіксованих $\alpha,\ \beta,\ \gamma$ базова мережа була масштабована з різними φ.

Для оптимального співвідношення точності та затраченого часу було вирішено використати в роботі мережі EfficientNet-B5 (рис. 14) та EfficientNet-B6 (рис. 15). Їх архітектури в минулих позначеннях виглядають так:

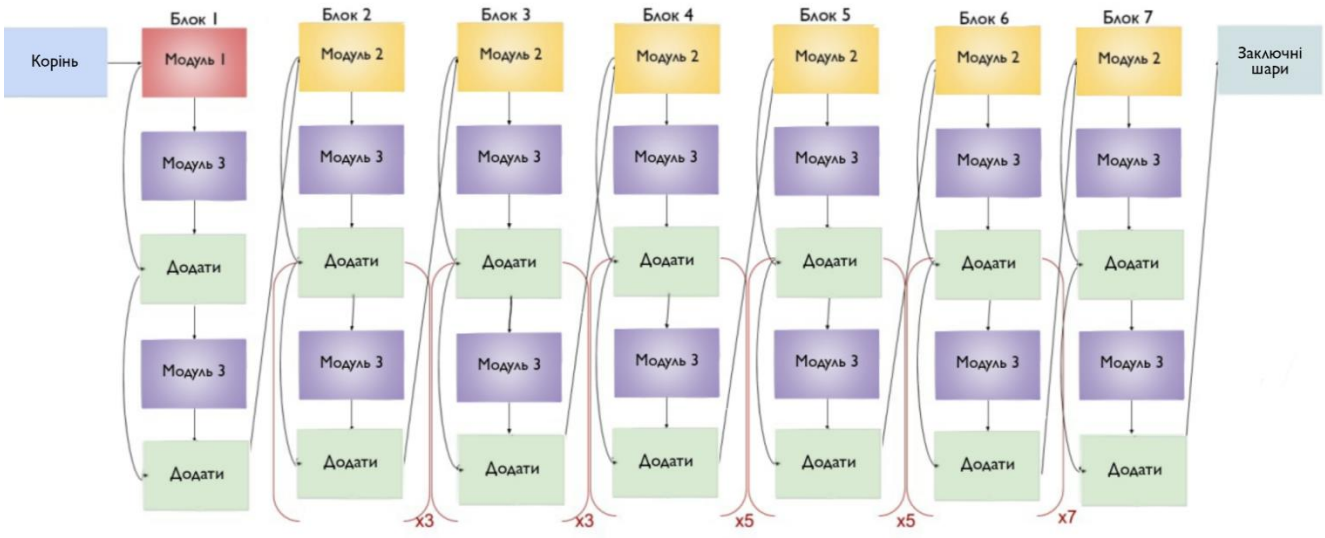

Рисунок 14 – Архітектура EfficientNet-B5, вихідна кількість каналів – 2048

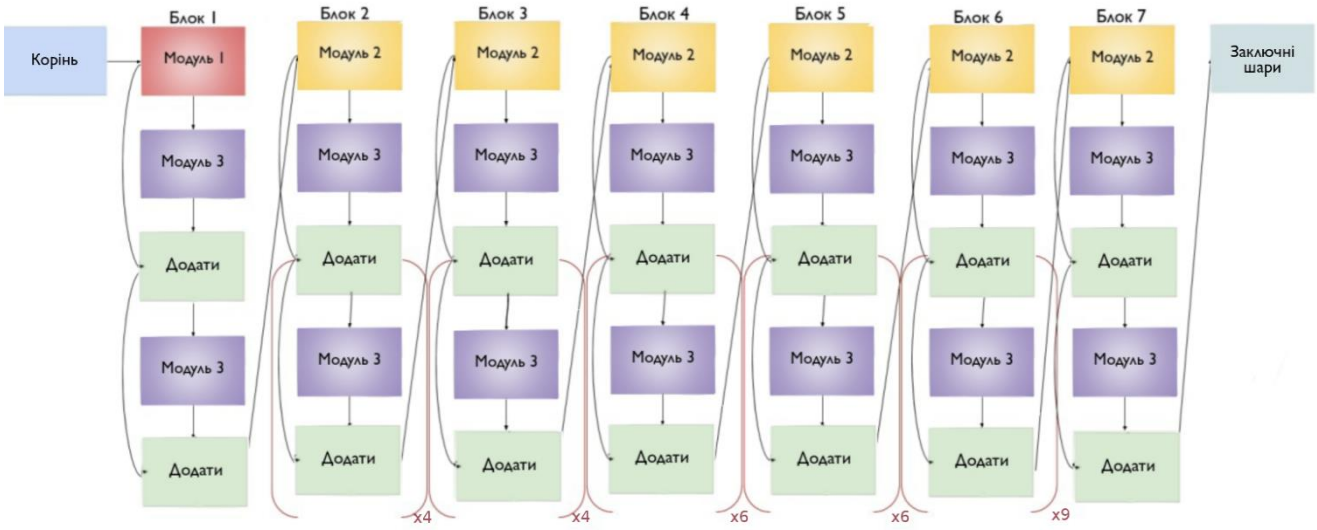

Рисунок 15 – Архітектура EfficientNet-B6, вихідна кількість каналів – 2304

2.4. Адитивна кутова втрата відступу (ArcFace)

Адитивна кутова втрата відступу (*ArcFace*, або Additive Angular Margin Loss) – це функція втрат, що була представлена в [17] і використовується в задачах розпізнавання (зазвичай облич). Вона була розроблена з метою досягти двох цілей.

1. Компактність усередині класу – додання екземплярів ближче до відповідного позитивного центру.

2. Розбіжність між класами – додання екземплярів далі від негативних центрів.

Функція *ArcFace* перетворює цільові логіти (ненормалізовані передбачення моделі) як $W_j^T x_i = \| W_j \| \| x_i \| \cos\theta_j$, де θ_j – кут між вагою W_j (останнього шару) та ознакою x_i ($\theta_j = \arccos(W_j^T x_i)$). Також зміщення останнього шару покладається нулем: $b_j = 0$. Далі вага $\| W_j \| = 1$ фіксується за допомогою нормалізації, а глобальна ознака $\| x_i \|$ фіксується l_2 нормалізацією та масштабована до s. Цей крок нормалізації забезпечує те, що передбачення залежить лише від кута між ознакою та вагою, а глобальні ознаки розподілені на гіперсфері радіуса s. Для того щоб одночасно покращити компактність усередині класу та відстань між різними

класами, додається адитивний кутовий відступ m між x_i та W_j. Функція втрат *ArcFace* визначається так:

$$\mathcal{L}_{\text{ArcFace}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp\{s\cos(\theta_{y_i,i}+m)\}}{\exp\{s\cos(\theta_{y_i,i}+m)\}+\sum_{j\neq y_i}\exp\{s\cos(\theta_{j,i})\}}$$

Класифікаційна границя:

$$\| x \| (\cos(\theta_1 + v) - \cos\theta_2) = 0$$

При кутовій відстані між двома центрами класів θ максимальною відстанню між вхідним даним та центром відповідного класу буде $\theta_1 = \frac{\theta - v}{2}$.

Геометрично різницю між функціями втрат м'якого максимуму з перехресною ентропією (*Softmax + Cross Entropy*) та *ArcFace* проілюстровано на рис. 16:

а) Softmax + Cross Entropy

б) ArcFace

Рисунок 16 – Приклад для функцій втрат для 8 об'єктів у двомірному просторі ознак: а) функція м'якого максимуму з перехресною ентропією; б) адитивна кутова втрата відступу

Як видно, хоч *Softmax + Cross Entropy* і дає відокремлювальні класи, на їх межах прийняття рішення є важким через неоднозначність. Разом з тим *ArcFace* розподіляє класи на очевидній відстані один від одного, зберігаючи компактність у

межах класу. Інша проблема *Softmax + Cross Entropy*, що кількість ваг на останньому повнозв'язному шарі моделі лінійно збільшується з кількістю класів, ускладнюючи навчання, не поширюється на *ArcFace*.

2.4.1. ArcFace з підцентрами класів

Модифікація функції втрат *ArcFace*, що була запропонована в [18], базується на ідеї впровадження в кожному класі підцентрів. Якщо зображення зашумлене (таким шумом може вважатися, наприклад, інший ракурс), то воно не є частиною відповідного позитивного класу, що впливає на навчання моделі, адже *ArcFace* генерує доволі значну втрату.

ArcFace з підцентрами послаблює компактність усередині класу, вводячи підкласи, що дає свободу екземпляру бути близьким до будь-якого з позитивних підцентрів, а не тільки одного. Один з підкласів є домінантним та містить більшість екземплярів, що розпізнаються просто, тоді як недомінантні підкласи містять зашумлені зображення або ті, що складніше розпізнати. Завдяки цьому можна підвищити точність класифікації та якість глобальних ознак, роблячи модель більш стійкою до шуму.

Не зважаючи на те, що наявність підкласів може підвищити стійкість, вона може понизити якість компактності всередині класу. Через це спочатку модель навчається на вибірці без зашумлених екземплярів, і вони додаються після того, як модель навчиться впевнено розрізняти класи.

Функція втрат *ArcFace* з підцентрами класів задається таким чином:

$$\mathcal{L}_{\mathrm{ArcFace}_{\mathrm{subcenter}}} = -\log \frac{e^{s\cos\left(\theta_{i,y_i}+m\right)}}{e^{s\cos\left(\theta_{i,y_i}+m\right)} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos\theta_{i,j}}},$$

де

$$\theta_{i,j} = arccos(\max_k (W_{j_k}^T \mathrm{x}_i)), k \in \{1, \dots, K\}.$$

2.4.2. ArcFace з мінливими відступами

Крім фіксованого m (відступу), можна змінювати його залежно від розміру класу. Для класів, що складаються з меншої кількості представників у навчальній вибірці, вірогідніше помилитися при верифікації та некоректно зробити вибір. Таким чином, для малих класів важливо перебувати на більшій відстані від усіх інших.

Візьмемо таку залежність між розміром класу та відступом для нього:

$$f(n) = a \cdot n^{-\lambda} + b,$$

де $a,\ b,\ \lambda$ – параметри. λ>0 відповідає за форму функції, а $a,\ b$ – найменше та найбільше можливе значення.

2.5. Узагальнена середня агрегація

Було показано [19], що в згорткових нейронних мережах згорнені ознаки, отримані на дальніх шарах, є вичерпним описом всього зображення, що включає як низькорівневі, так і більш абстрактні ознаки. Вони мають достатні дискримінаційні здатності та можуть бути агреговані, що дає простішу архітектуру та підвищення продуктивності.

Уведемо поняття максимальної, середньої та узагальненої середньої агрегації.

При такому вхідному зображенні вихідним сигналом згорткової мережі є тривимірний тензор форми $K \times H \times W$, де K – кількість каналів, H – висота карти ознак, а W – ширина карти ознак. Якщо $\mathcal{X}_k$ представляє активацію просторової карти об’єктів $H \times W$, то мережа складається з K таких карт об’єктів.

Максимальна агрегація:

$$\mathbf{f}^{(m)} = [\mathrm{f}_1{}^{(m)} \dots \mathrm{f}_k{}^{(m)} \dots \mathrm{f}_K{}^{(m)}]^\top, \quad \mathrm{f}_k{}^{(m)} = \max_{x \in \mathcal{X}_k} x.$$

Для кожної карти $\mathcal{X}_k$ беремо максимальне значення, щоб отримати векторне представлення зображення, що має довжину K.

Середня агрегація:

$$\mathbf{f}^{(a)} = [\mathrm{f}_1{}^{(a)} \dots \mathrm{f}_k{}^{(a)} \dots \mathrm{f}_K{}^{(a)}]^\top, \quad \mathrm{f}_k{}^{(a)} = \frac{1}{|\mathcal{X}_k|} \sum_{x \in \mathcal{X}_k} x$$

Для кожної карти $\mathcal{X}_k$ беремо середнє значення, щоб отримати векторне представлення зображення, що має довжину K.

Узагальнена середня агрегація:

$$\mathrm{f}^{(g)} = [\mathrm{f}_1{}^{(g)} \dots \mathrm{f}_k{}^{(g)} \dots \mathrm{f}_K{}^{(g)}]^\top, \quad \mathrm{f}_k{}^{(g)} = \left(\frac{1}{|\mathcal{X}_k|} \sum_{x \in \mathcal{X}_k} x^{p_k} \right)^{\frac{1}{p_k}}.$$

Таким чином, максимальна (при $p_k \to \infty$) та середня (при $p_k = 1$) агрегації є частковими випадками узагальненої середньої агрегації. Вектор ознак має одне значення для кожної карти ознак, і має розмірність К. Агрегаційний параметр p_k може бути як і гіперпараметром моделі, так і вивчений під час навчання, адже агрегування є частиною зворотнього поширення помилки.

РОЗДІЛ 3 ВИДІЛЕННЯ ЛОКАЛЬНИХ ОЗНАК

3.1. Автокодувальник

Автокодувальником називається тип нейронної мережі, що навчається відтворити вхідне дане у своєму виході, зазвичай для задачі зменшення розмірності. Така мережа складається з двох частин (рис. 17): кодувальника, що відображає вхідне дане в код, та декодувальника, що реконструює вхідне дане з коду. Код описується за допомогою внутрішнього шару мережі.

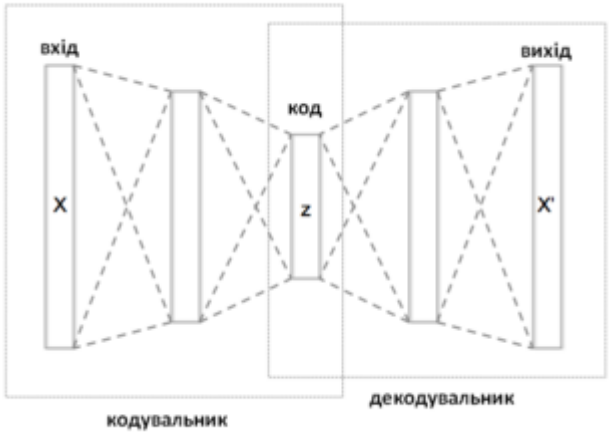

Рисунок 17 – Структура автокодувальника

Автокодувальники належать до моделей навчання без вчителя, а їх архітектура це зазвичай не-рекурентна нейронна мережа прямого поширення.

Визначимо кодувальник та декодувальних як переходи ϕ та ψ відповідно, де:

$$\phi: \mathcal{X} \to \mathcal{F}$$
$$\psi: \mathcal{F} \to \mathcal{X}$$
$$\arg\min_{\phi,\psi} \|\mathcal{X} - (\psi \circ \phi)\mathcal{X}\|^2$$

За наявності лише одного внутрішнього шару вхід $\mathbf{x} \in \mathbb{R}^d$ відображається автокодувальником на $\mathbf{z} \in \mathbb{R}^p$, що називається кодом:

$$\mathbf{z} = \sigma_1(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

де σ – функція активації.

Потім $\mathbf{z}$ відображається на $\mathbf{x}'$, що має однакову форму з $\mathbf{x}$ та називається відбудовою:

$$\mathbf{x}' = \sigma_2(\mathbf{W}'\mathbf{z} + \mathbf{b}').$$

Функція втрат такої мережі повинна прагнути до нуля при наближенні відбудови до вхідного даного, тому використовується метрика l_2:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma_2(\mathbf{W}'(\sigma_1(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2$$

У випадку, коли розмірність $\mathcal{F}$ (простір ознак) є меншою за розмірність простору $\mathcal{X}$ (вхідний простір), можна говорити про вектор ознак $\phi(x)$ як про стиснене представлення входу x.

3.2. Механізм уваги

Увага – це техніка, що використовується в машинному навчанні для виділення важливих частин вхідних даних та послаблення всіх інших. Тобто більша кількість обчислювальної потужності має бути виділена на ті частини, що є важливими, хоч і невеликими.

Зазвичай механізм уваги використовується після роботи основної мережі, що виділяє ознаки і допомагає відкинути менш доречні ознаки для поставленої задачі для поліпшення обчислювальної ефективності.

Нехай $\mathbf{f}_n \in \mathrm{R}^d$, $n = 1,\ldots, N$ – це d-розмірні ознаки, що були вивчені основною мережею. Тоді задачею механізму уваги є вивчення оціночної функції $\alpha(\mathbf{f}_n; \theta)$ для кожної з ознак, де θ – параметри. На виході, взявши зважену суму векторів ознак, отримуємо логіт $\mathbf{y}$:

$$\mathbf{y} = \mathbf{W}\left(\sum_n \alpha(\mathbf{f}_n; \theta) \cdot \mathbf{f}_n\right),$$

де $\mathbf{W} \in \mathrm{R}^{M \times d}$ – ваги останнього повнозв'язного шару основної мережі.

Як функція втрат вибирається крос-ентропія:

$$\mathcal{L} = -\mathbf{y}^* \cdot \log\left(\frac{\exp(\mathbf{y})}{\mathbf{1}^{\mathrm{T}} \exp(\mathbf{y})}\right),$$

де **y***– коректний клас, а **1** – одиничний вектор.

У результаті тренування отримуємо матрицю оціночних функцій релевантності.

3.3. Співставлення локальних ознак

За наявності двох зображень, для кожного з яких відомі локальні ознаки, виникає задача їх співставлення для визначення, чи один і той самий об’єкт наявний на цих зображеннях. Для кожної з ознак можна знайти найближчу (в даному просторі при даній метриці) ознаку з іншого зображення, таким чином співставивши їх та утворивши пару. Проте таким чином не можна гарантувати, що всі пари точно є коректними, отже, треба ще й співставити координати. Очевидно, усі світлини зроблені з різних точок (рис. 18), тому координати локальних ознак на зображеннях не будуть співпадати.

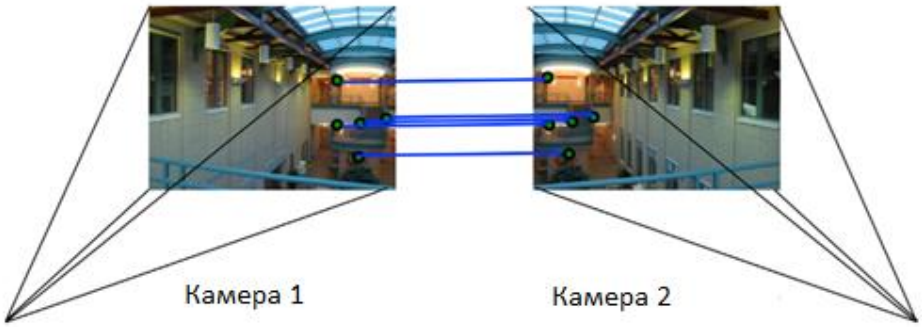

Рисунок 18 – Відповідні локальні ознаки на світлинах, зроблені з різних точок

Для визначення коректності відповідностей між локальними ознаками на зображеннях використовується проективне перетворення (рис. 19) та консенсус довільної вибірки.

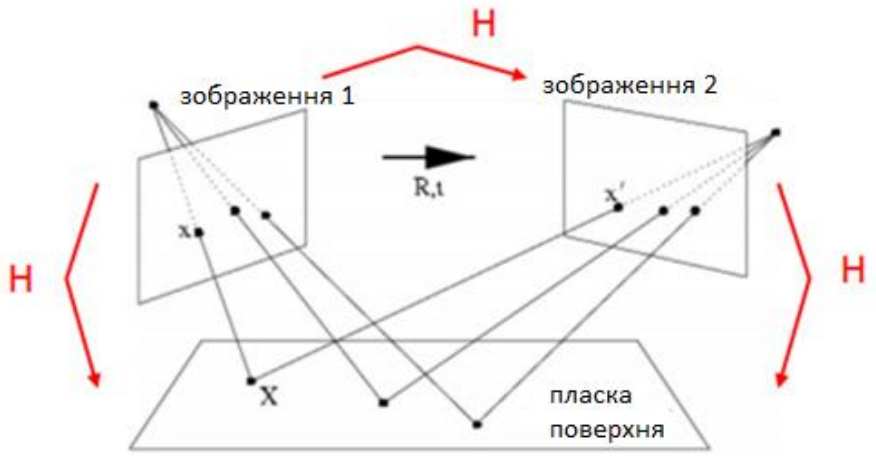

Рисунок 19 – Візуалізація проективного перетворення H

3.3.1. Консенсус довільної вибірки (RANSAC)

Консенсус довільної вибірки (RANSAC, RANdom SAmple Consensus) [4] – ітеративний метод для оцінки, які дані з набору спостережуваних є викидами для визначення параметрів деякої математичної моделі. Припускається, що всі дані можна поділити на викиди, що є результатами шумів та не задовольняють певній моделі, та не-викиди, що цій моделі задовольняють.

RANSAC для оцінки проективного перетворення між зображеннями p_1, p_2:

1. Випадково вибрати чотири пари локальних ознак.
2. Обрахувати точне проективне перетворення H.
3. Обрахувати не-викиди, де сума квадратів різниць між p_1, Hp_1 менша за певне порогове значення.
4. Повторити кроки 1–3.
5. Залишити найбільший набір не-викидів.
6. Перерахувати оцінку найменших квадратів H для всіх не-викидів.

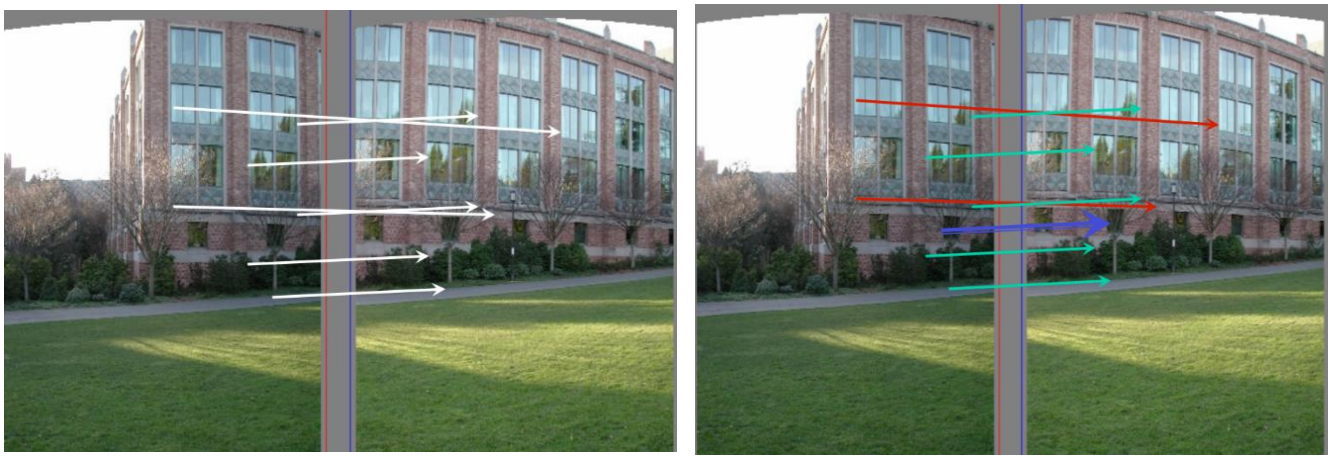

Рисунок 20 – Для набору всіх пар локальних ознак (зліва) було використано метод RANSAC. Пари, позначені зеленим, склали більшість, і тому є не-викидами, тоді як червоні пари є викидами. Синім кольором позначено усереднений вектор перетворення

Беручи за основу RANSAC, існують й інші алгоритми для співставлення локальних ознак. Зокрема, DEGENSAC [20] має таку ж саму обчислювальну складність, але знаходить набагато більше відповідностей між локальними ознаками, ніж RANSAC.

РОЗДІЛ 4 НАВЧАННЯ МЕРЕЖІ

4.1. Збереження даних

4.1.1. K-розмірне дерево

K-розмірне дерево – це структура даних, запропонована в [21], що використовується для упорядкування точок в k-мірному просторі. Це бінарне дерево, яке має k-мірну точку в кожному листі, а всі інші вузли неявно породжують відокремлюючу гіперплощину, що ділить простір на два напівпростори. Кожен із вузлів асоціюється з одним з k вимірів з відповідною гіперплощиною, що перпендикулярна цьому виміру. Приклад 3-розмірного дерева наведений на рис. 21.

K-розмірне дерево може бути побудоване за $O(n(k+\log(n)))$. Вимога по пам’яті: $O(kn)$.

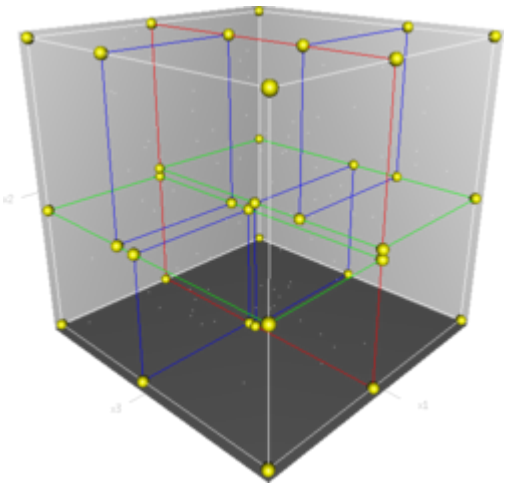

Рисунок 21 – 3-розмірне дерево

4.1.2. Пошук найближчого сусіда в k-розмірному дереві

Для знаходження в дереві точки (точок), що є найближчою до даної, застосовується алгоритм пошуку найближчого сусіда. Властивості дерева допомагають зробити це ефективно, оскільки великі частини простору можна легко відкинути.

Алгоритм:

1. Початок: кореневий вузол. Рухаємось по дереву рекурсивно, йдучи (умовно) ліворуч чи праворуч залежно від співвідношення точки до поточного вузла: якщо вона менша чи більша, йдемо у відповідну сторону розділеної розмірності.
2. При досягненні листа розраховується відстань від точки до нього, і зберігаємо цю відстань як найкращу на даний момент.
3. Розвиваємо рекурсію, виконуючи такі дії на кожному вузлі:
    a. Вузол стає найкращим на даний момент, якщо він знаходиться ближче найкращого на даний момент.
    b. Перевіряємо, чи можуть бути листи, що знаходяться ближче, ніж найкраща на даний момент точка, з іншої сторони роздільної площини. Для цього виконується перетин роздільної гіперплощини з гіперсферою навколо точки пошуку, де за радіус гіперсфери береться найкраща відстань на даний момент. Порівнюючи відстань між координатою розбиття точки пошуку та поточним вузлом та відстань від точки пошуку до найкращої на даний момент, отримуємо відповідь, оскільки гіперплощини вирівняні по осі.
        i. Якщо гіперсфера та площина перетинаються, то з її іншої сторони можуть знаходитися точки, що є ближчими до заданої. Алгоритму треба продовжувати йти по іншій гілці під поточного вузла, відповідно до процесу, описаного вище.
        ii. Якщо гіперсфера та площина не перетинаються, то алгоритм піднімається на один вузол вище по дереву, а інший бік вузла не розглядається.
4. При досягненні та розгляді кореня алгоритм завершує роботу.

Середня кількість переглянутих елементів: $O(h)\cdot(O(\log(h)+1))$, де h – висота дерева.

Модифікація алгоритму для пошуку k найближчих сусідів доволі проста – треба зберігати k найкращих відстаней на даний момент, а також відсікати гілки лише тоді, коли k точок вже було знайдено.

4.2. Схема навчання

Для реалізації було використано ідею методу навчання DELG [8], що поєднує навчання глобальних та локальних дескрипторів в одній основній моделі, яку можна бачити на рис. 22. На відміну від інших моделей, де ознаки вивчаються окремо, це значно пришвидшує навчання, і не погіршує результат, адже процес навчання є схожим для обох із них.

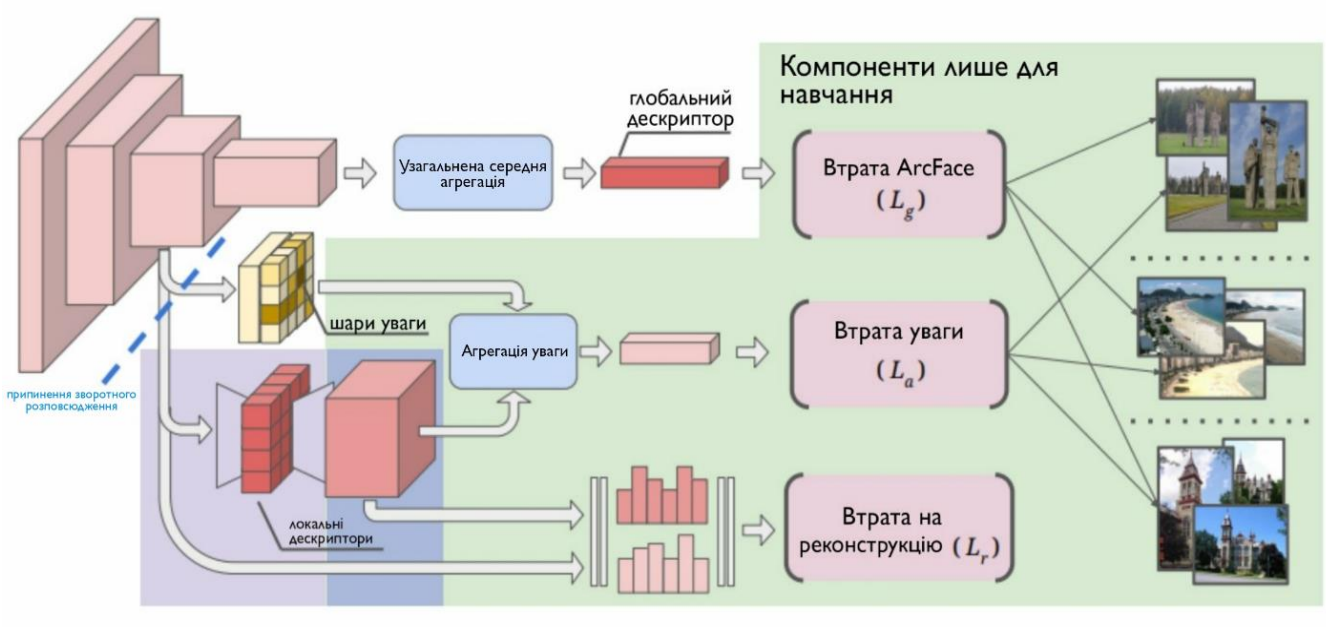

Рисунок 22 – Загальний метод навчання мережі

Модель навчання складається з декількох етапів.

1. Основна нейронна мережа, для якої функцією втрат є ArcFace з підцентрами та мінливими відступами (для зворотного розповсюдження помилки). Було натреновано три глибокі мережі: EfficientNet-B5, EfficientNet-B6 та ResNet 152 V2.

2. Глобальні ознаки:

До останнього шару (де C – кількість каналів) основної мережі застосовується узагальнена середня агрегація, перетворюючи матрицю з кожного каналу на число, утворюючи глобальний дескриптор довжини C. Після застосування функції втрат

до отриманого вектора, виконується метод зворотного розповсюдження помилки на основній нейронній мережі.

3. Локальні ознаки:

a. Передостанній шар основної мережі (кількість каналів – C_2) передається в нейронну мережу, що вчиться виокремлювати регіони для механізму уваги. Вихідним шаром такої мережі є матриця розміру вхідного шару, але з одним каналом. Функцією втрат такої мережі (втрата уваги) є функція максимальної м'якості з перехресною ентропією.

b. Передостанній шар основної мережі передається в автокодувальник з метою зменшення розмірності для того, щоб позбавитися від глобальних ознак та залишити лише локальні. Мережа являє собою одношарову згорткову мережу. Функцією втрат такої мережі є поєднання функції максимальної м'якості з перехресною ентропією та середньоквадратичної помилки.

c. Результат роботи декодувальника, що намагається відтворити вхідний шар, поелементно перемножується з матрицею уваги для кожного каналу під час агрегації уваги. Після застосування функції втрат до отриманого вектора розміру C_2 виконується метод зворотного розповсюдження помилки на вище описаних мережах.

Після тренування глобальні дескриптори зображень з набору тренування додаються до бази у вигляді k-розмірного дерева. Оскільки тренувалось декілька основних мереж, то отримані глобальні дескриптори конкатенуються.

4.3. Розподілене навчання

4.3.1. Збереження даних у «Шардах»

Набір даних Google Landmarks v2 було розділено на 2048 шардів, тобто приблизно 2441 зображень у кожному шарді. Кожен шард зберігається у форматі TFRecords – це спеціальний формат для зчитування та обробки даних у фреймворці TensorFlow. Розподілення даних на якомога більше шардів – дуже важливий етап. Це дає змогу завантажувати дані паралельно, що значно збільшує утилізацію тензорних процесорів TPU v3.

4.3.2. Схема розподіленого навчання

Щоб навчити найпотужніші нейронні мережі (Efficient Net B5, Efficient Net B6 та ResNet 152 V2) на повному наборі даних Google Landmarks v2 (що містить понад 5 мільйонів зображень) було створено унікальну схему розподіленого навчання (тренування) на хмарних сервісах Google Cloud Services. Повну схему навчання і валідації можна побачити на рис. 23.

Розподілене тренування нейронної мережі складається з таких частин.

1. На початку кожної епохи навчання (всього понад 50 епох для кожної нейронної мережі) сервіс балансування навантажень робить випадковим порядок шардів даних та для кожного шарду даних вирішує, до якого TPU v3 блоку вона потрапляє та в якому порядку.

2. Шарди даних, що знаходяться на Google Cloud Storage Bucket в одному з найбільших дата-центрів компанії Google в локації «us-central-1a», через інтернет мережу передаються в сервіс балансування даних.

3. Сервіс балансування перенаправляє шарди даних до відповідного TPU v3 блоку, що знаходиться в TPU v3-8 кластері.

4. Дані передаються потоками. У кожному блоці TPU дані завантажуються та обробляються у 4-х CPU паралельно, окремо від процесу навчання нейронної мережі. Це робиться для того щоб TPU v3 блок не очікував завантаження всього шарду цілком, а міг одразу почати навчання.

5. Обчислюються градієнти для реплікації нейронної мережі відносно цільової функції на кожному з окремих блоків TPU v3.

6. Після цього збираємо ці градієнти у спільних процесорах TPU v3-8 кластера та агрегуємо їх у нашу нейронну мережу. Робиться дзеркальна реплікація мережі, та кожну реплікацію посилаємо до відповідного TPU блоку.

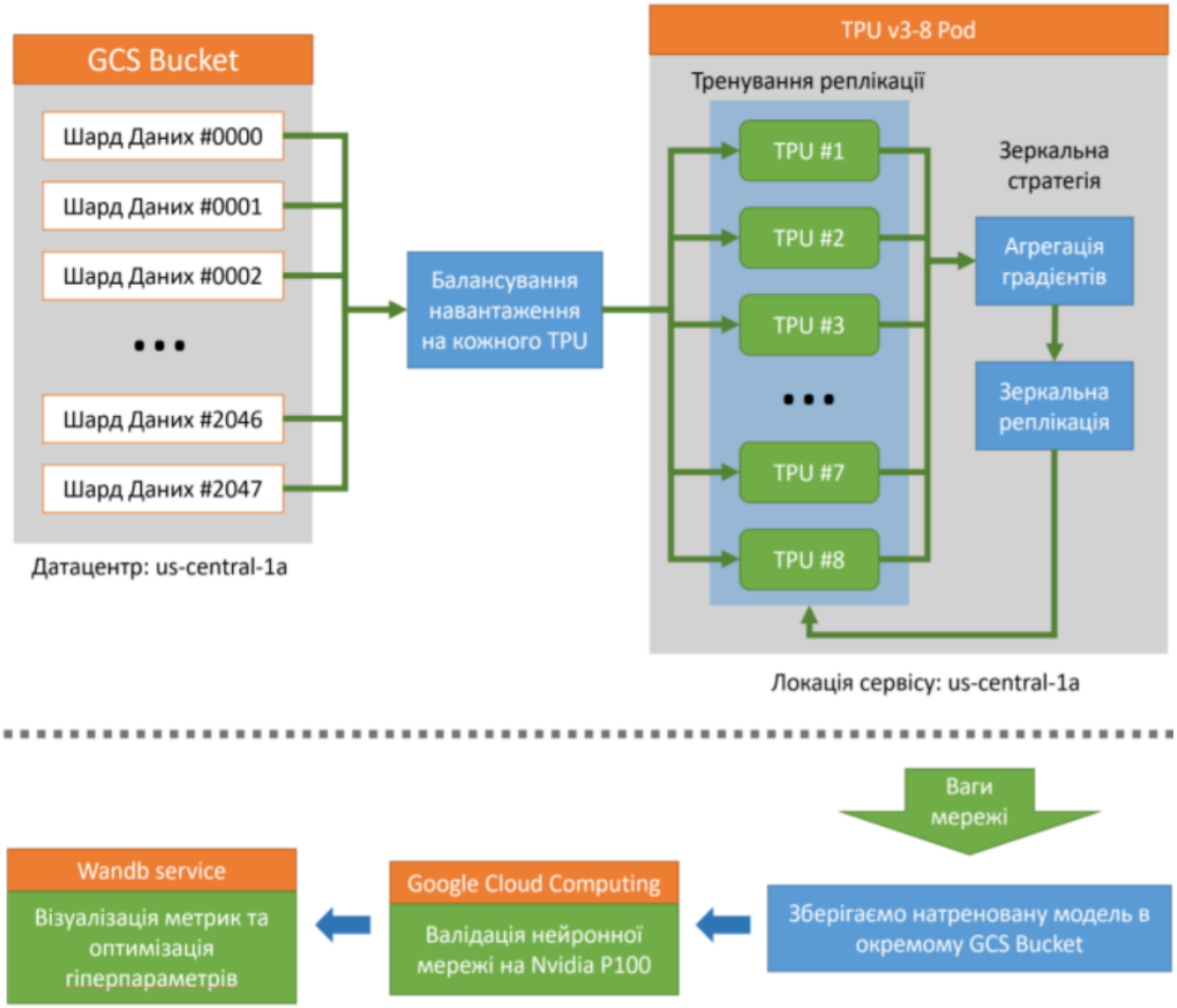

Рисунок 23 – Схема розподіленого навчання

Після кожної епохи натреновані ваги мережі зберігаються на окремому GCS Bucket. Після цього запускається валідація мережі незалежно від процесу навчання, а саме:

1. Проганяємо нейронну мережу по набору даних для валідації на окремій інстанції сервісу обчислень компанії Google, а саме на інстанції з прискорювачем тензорних обчислень Nvidia P100. Обчислюємо метрики точності.
2. Візуалізація метрик, локальних ознак та динаміки навчання мережі робиться на сервісі Wandb (Weights and Biases).
3. Також аналізуємо результати навчання для подальшої оптимізації гіперпараметрів (тобто параметрів навчання нейронної мережі).

Локацію датацентру для зберігання даних та локації обчислювальних кластерів TPU v3-8 та P100 було обрано таким чином, щоб вони перебували якомога ближче один до одного, щоб зменшити затримки мережі Інтернет.

4.3.3. Балансування навантажень

Алгоритм балансування дуже простий. Випадковим чином перетасовуються шарди даних. Потім ці шарди рівномірно розподіляються для кожного блоку TPU v3.

4.3.4. Тензорний Процесор (TPU v3)

Тензорний процесор (англ. tensor processing unit, TPU) – це інтегральна схема специфічного застосування (ASIC), призначена для прискорення розрахунків штучного інтелекту, розроблена компанією Google.

Кожен блок TPU v3 має обчислювальну потужність 52.5 терафлопсів (TFLOPS). Отже, маючи 8 таких блоків, обчислювальна потужність кластера TPU v3-8 становить 420 терафлопсів.

Для порівняння, найпопулярніший прискорювач Nvidia GeForce 2080ti, що використовують для тренування нейронних мереж, має потужність лише 13.4 терафлопсів. Найпопулярніші серверні прискорювачі Nvidia Tesla P100, яку використовували самі автори публікації DELG (вони тренували паралельно на 32 таких прискорювачах), мають потужність 18.7 терафлопсів. Найшвидший прискорювач Nvidia Tesla V100 має потужність 112 терафлопсів. Візуально порівняння зображено на рис. 24.

Прискорювач	Nvidia Tesla P100 (Pascal)	Nvidia Tesla V100 (Voltra)	TPU v3-8
Потужність	18.7 TFLOPS	112 TFLOPS	420 TFLOPS
Пам'ять	16 GB	32 GB	128 GB

Рисунок 24 – Порівняння прискорювачів

4.3.5. Навчання на одному блоці TPU v3

На кожному блоці TPU v3 навчання реплікації нейронної мережі відбувається звичайним чином, подібно тому, як навчаються на звичайних прискорювачах (наприклад Nvidia P100). Шарди даних із датацентру завантажуються паралельно на чотирьох потоках локального процесора (тобто одночасно, один блок TPU v3 завантажує 4 різних шарди). Отримані дані потім передаються в тензорний процесор для навчання.

4.3.6. Стратегія агрегації та реплікації

Для передачі змінних оновлень на всі пристрої TPU v3 використовуються ефективні алгоритми All-Reduce. All-Reduce агрегує тензори з усіх пристроїв, додаючи їх, і робить доступними на кожному пристрої. Це злитий алгоритм, який є дуже ефективним і може значно зменшити накладні витрати на синхронізацію. Доступно багато алгоритмів і реалізацій All-Reduce, залежно від типу зв’язку між пристроями. У цій роботі було використано стандартну реалізацію All-Reduce, що вбудовано в TPU. Ця реалізація гарантує ефективність операцій All Reduce та інших колективних операцій на декількох блоках TPU.

4.4. Деталі навчання нейронних мереж

Для кожної з обраних архітектур (Efficient Net B5, Efficient Net B6 та ResNet 152 V2), незважаючи на те, що використовуємо спільний загальний метод навчання (DELG) та спільну цільову функцію ArcFace, було обрано різні параметри та деталі навчання.

4.4.1. Метод оптимізації

Було поставлено експерименти з різними методами оптимізації: Adam [22], AdaGrad, простий SGD та SGD з імпульсом Нестерова. Для кожного з цих методів оптимізації були також поставлено експерименти з різними значеннями кроку оптимізації та методами планування значення кроку оптимізації.

Як показують експерименти, Adam та AdaGrad показують нестабільні результати при довгочасному навчанні. Це відбувається тому, що в цих алгоритмів є відома проблема вибухання градієнтів унаслідок ділення на значення ковзної середньої, яка після декількох десятків мільйонів ітерацій стає дуже близькою до нуля. Цей ефект детально описано в роботі [23].

4.4.2. Планування значення кроку оптимізації

Одним із найголовніших факторів навчання нейронних мереж є вибір значення кроку оптимізації. Якщо це значення дуже велике, то нейронна мережа не зможе знайти точки глобального мінімуму та буде осцилюватись навколо точки мінімуму. Якщо поставити дуже маленьке значення, то нейронна мережа скоріш за все застрягне у точці локального мінімуму.

Найкращим на сьогодні рішенням цієї проблеми є метод циклічного планування значень кроку оптимізації з повторним запуском, що був запропонований у роботі [24], з покращеннями, запропонованими в роботі [25]. Порівняння можна побачити на рис. 25.

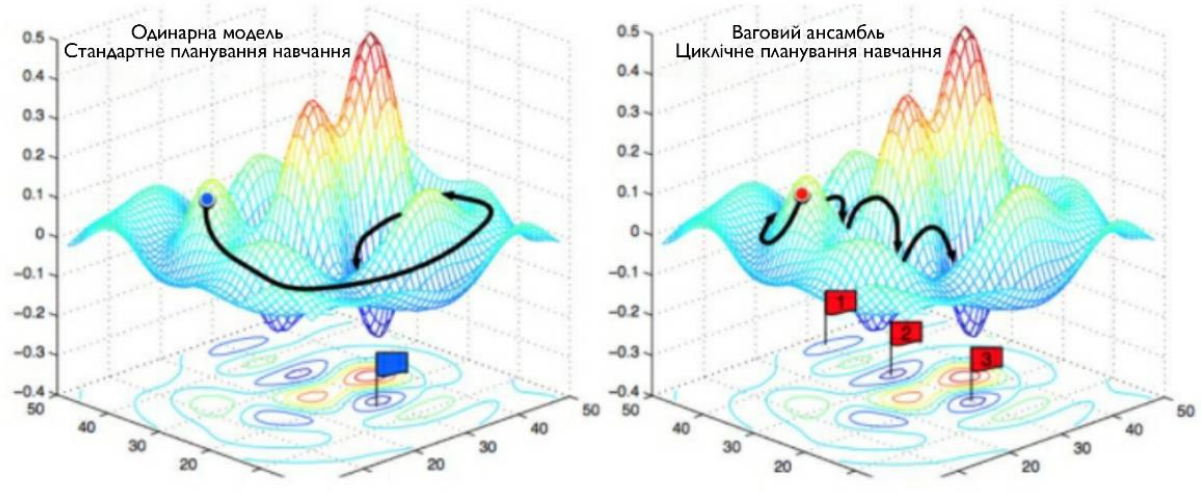

Рисунок 25 – Стандартне та циклічне планування кроку оптимізації

Такий метод дає змогу з більшою ймовірністю знаходити мінімуми гіперповерхні оптимізації, що є близькими до глобального мінімуму, як показано

на ілюстрації. Під кінець навчання робиться ваговий ансамбль із ваг найкращих моделей.

4.4.3. Планування значення відступу для ArcFace

Як показано в роботі [26], на початку навчання більш вигідно мати менші значення відступу для цільової функції ArcFace. Пізніше, коли нейронна мережа вже має певне наближення до оптимальних параметрів, можемо збільшувати значення відступу для отримання більш дискримінативних глобальних ознак.

Отже, під кінець кожного періоду циклічного планування значень кроку оптимізації збільшуємо абсолютний коефіцієнт значення відступу ArcFace. Більш детальне порівняння результатів з плануванням значення відступу та без нього наведено в розділі “Результати”.

4.4.4. Багатоступеневе тренування

Оскільки набір даних Google Landmarks Dataset v2 дуже шумний (більшість зображень було отримано внаслідок пошуків Google, без анотації людиною), навчання на всьому наборі даних дасть неоптимальні результати. Отже, дуже важливо мати якісне початкове наближення.

Тому навчання було виконано в 3 етапи:

1. Навчання на розмірі 512x512 на очищеному наборі даних з GLD2, що містить 1.6 мільйонів зображень та 81313 класів. Цей етап займає 10 епох.
2. Використовуючи натреновані ваги з 1-го етапу, використовуючи інші 4 мільйонів зображень та 203094 класів, продовжити навчання. Цей етап займає 5 епох.
3. Використовуючи натреновані ваги з 2-го етапу, продовжити навчання на повному наборі даних, поступово збільшуючи розмір вхідного зображення (10 епох на 640x640, 5 епох на 768x768, і потім 1 епоху на розмірі 1024x1024).

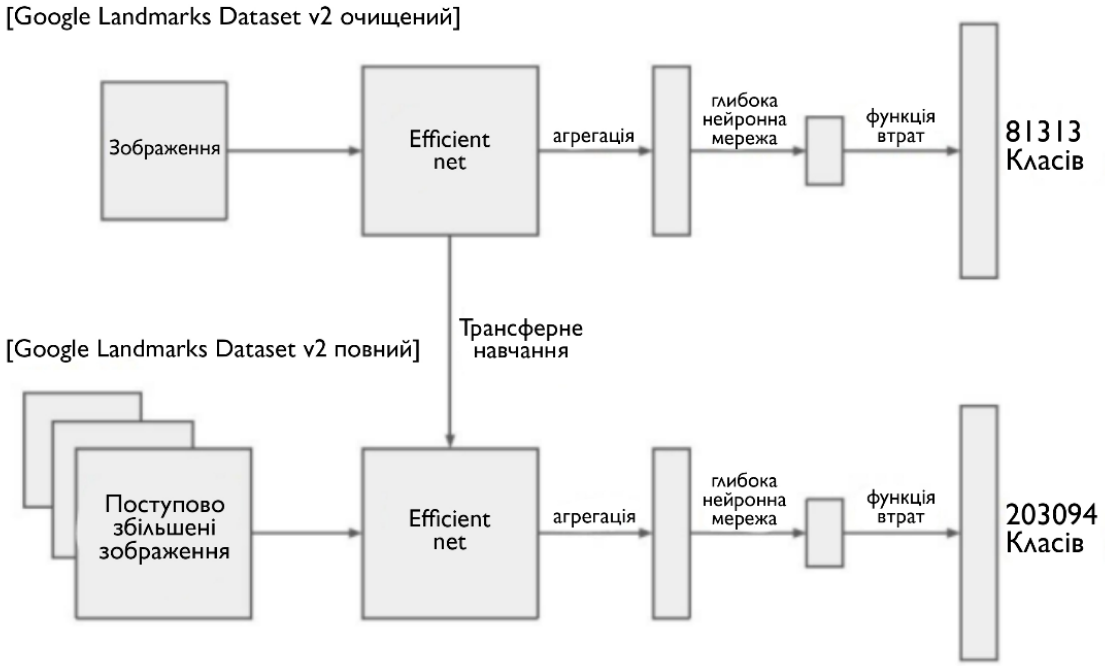

Рисунок 26 – Багатоступеневе навчання великої нейронної мережі на масштабних даних

Більш детальне порівняння результатів із багатоступеневим навчанням наведено в розділі “Результати” наприкінці цієї роботи.

4.4.5. Аугментація даних

Аугментація – це клас методів обробки та модифікації вхідного зображення при навчанні нейронної мережі. Це використовується для того, щоб комбінаторно збільшити варіацію вхідних зображень. Для аугметації було використано комбінації з таких методів:

- Дзеркальне відображення горизонтально чи вертикально.
- Випадкове обертання зображення.
- Випадкова зміна яскравості, контрасту та експозиції зображення.
- Випадкова гамма-корекція зображення.
- CutMix [27], де різні вхідні зображення комбінуються.

4.4.6. Відлуння даних (Data Echoing)

Оскільки дані для навчання перебувають на окремому дата-центрі та доступ до даних з блоків TPU маємо лише через мережу інтернет, було використано просту техніку відлуння даних (Data Echoing), запропоновану в [28]. Цей метод дає змогу більш ефективно використовувати локально доступні вхідні дані. Завдяки цьому методу було отримано прискорення в 20 % при навчанні нейронної мережі.

РОЗДІЛ 5 ЕКСПЕРИМЕНТИ ТА РЕЗУЛЬТАТИ

5.1. Схема обробки запиту

Для верифікації моделі було використано ідею методу верифікації DELG [8], яку можна побачити на рис. 27.

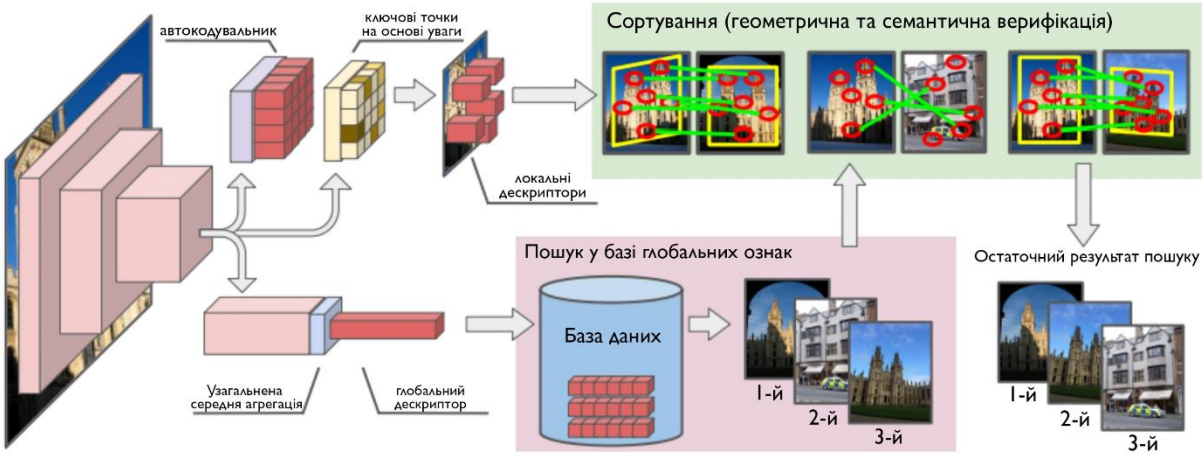

Рисунок 27 – Процес обробки запиту

При даному на вхід зображенні (запиту), пошук схожих зображень проводять так:

1. Зображення проганяється через основну натреновану нейронну мережу.
2. До останнього шару основної мережі застосовується узагальнена середня агрегація для отримання глобального дескриптора. Після цього в базі глобальних дескрипторів шукається m (m = 30) найближчих сусідів у k-розмірному дереві, і зображення, що відповідають цим дескрипторам, повертаються.
3. До передостаннього шару основної мережі застосовується кодувальна половина автокодувальника, що зменшує його розмірність за кількістю каналів. Після цього за допомогою механізму уваги виділяються ті локальні ознаки, що важливіші за інші. Той самий процес відбувається з усіма *m* зображеннями, що повернула база глобальних ознак.
4. Далі проводиться геометрична верифікація. До найкращих (за «уважністю», яка їм надана) векторів локальних ознак із заданого зображення

знаходяться найближчі вектори локальних ознак із зображень, повернутих із бази. З них викидаються сторонні за допомогою DEGENSAC. Якщо кількість ознак, що залишилася, більше порогового значення, зображення залишається. Ці зображення сортуються по кількості співпадінь ознак та повертаються як результат роботи моделі.

Варто зазначити, що насправді на вхід подається не одне зображення, а три його варіанти в різній розмірності. Глобальні ознаки беруться усереднені, а локальні ознаки виходять більш щільними.

5.2. Загальна середня точність

Загальна середня точність (ЗСТ) [29] – метрика, що вираховується для набору пар передбачення та оцінки впевненості, що відсортовано в низхідному порядку відносно оцінки впевненості.

Вираховується такою формулою:

$$\text{ЗСТ} = \frac{1}{M}\sum_{i=1}^{N} P(i)rel(i),$$

де:

- N – загальна кількість передбачень,
- M – загальна кількість вхідних зображень з хоча б одною пам’яткою на ньому,
- $P(i)$ – точність на позиції i,
- $rel(i)$ – релевантність i-го передбачення: 1 – якщо i-те передбачення коректне, інакше – 0.

5.3. Експерименти для підбору гіперпараметрів

По-перше, щоб порівняти різні методи навчання моделі для пошуку схожих зображень, а також для чесного порівняння з найкращими на цей момент опублікованими результатами від [8], зробимо детальне вивчення абляції за всіма параметрами. Візьмемо ЗСТ як основну метрику для порівняння різних моделей.

Для всіх вимірювань результатів додатковим обмеженням є те, що валідація повинна займати не більш ніж 9 годин на наборі даних для валідації, або не більш ніж 1 секунди на кожний запит пошуку схожих зображень.

Слід зазначити, що гіперпараметри, що підібрані у цьому розділі, були підібрані виключно на наборі даних для валідації. Результат метрики ЗСТ на наборі даних для тестування буде наведено лише для підтвердження ефективності методу та кореляції результатів на валідаційному наборі та тестовому.

5.3.1. DELG модель та параметри PyDegensac

Спочатку візьмемо модель натренованого автором методу DELG [8] (архітектуру якого ми взяли за основу), що доступний у відкритому доступі [30]. Візьмемо параметри локальної верифікації за допомогою PyDegensac та спробуємо їх покращити. А саме будемо розглядати такі параметри:

- Кількість невикидів, що розглядаємо.
- Довірчий поріг невикидів.
- Похибка репроекції [31] PyDegensac.

Результати викладені у табл. 2. Можна побачити, що різні параметри верифікації за допомогою PyDegensac можуть значно вплинути на кінцевий результат. Очевидно, збільшення кількості ітерації RANSAC дає більш точні результати, тому що ми ставимо більше евристичних експериментів.

Таблиця 2 – Порівняння гіперпараметрів геометричної верифікації за допомогою DEGENSAC

Ітерації RANSAC	Невикиди	Довірчий поріг	Похибка проекції	Валідація ЗСТ	Тестування ЗСТ
1’000	20	0.99	4.0	0.4426	0.4241
10’000	20	0.99	4.0	0.4830	0.4562
100’000	20	0.90	4.0	0.4824	0.4578
100’000	20	0.99	6.0	0.4809	0.4573
10’000’000	35	0.99	6.0	**0.4837**	**0.4609**

Вплив похибки проекції достатньо неочевидний. На набору даних для валідації збільшення цього параметру дає гірші значення метрики. На наборі даних для тестування це дає кращі метрики. Це свідчить про те, що набір даних Google Landmarks Dataset V2 має не так багато локальних ознак, що можна однозначно ідентифікувати.

Збільшення кількості ітерації RANSAC є дуже непрактичним способом. По-перше, це дає тільки відносно незначні покращення, тобто результат більш залежить від якості навченої моделі нейронної мережі. По-друге, при 10 мільйонів ітерації це займає дуже багато часу. Тому для подальших результатів будемо використовувати такі фіксовані значення:

- Кількість ітерації RANSAC: 100’000.
- Максимальна кількість невикидів: 20.
- Максимальна похибка репроекції: 4.
- Довірчий поріг проективного перетворення: 0.99.

Такий набір параметрів геометричної верифікації дає достатньо гарні результати та не займає багато часу для обчислення.

5.3.2. Вплив семантичної верифікації

Перед тим як перейдемо до результатів навчання власне нейронних мереж (ResNet 152 V2 та Efficient Net B6, B5) за допомогою покращень у методі навчання перевіримо вплив семантичної верифікації, тобто одне з наших покращень над роботою DELG [8], що ми взяли за основу. Будемо використовувати такі моделі для семантичної детекції:

- MobileNet + SSD (тобто, узявши MobileNet [32] навченої на ImageNet за основу, та використовуючи архітектуру для детекції об’єктів SSD [33]) навченої на наборі даних Open Images V4 [34].
- Inception-Resnet V4 + Faster R-CNN (тобто, узявши Inception-ResNet V4 [35] навченої на ImageNet за основу, та використовуючи архітектуру Faster R-CNN [36] для детекції об’єктів), навченої на Open Images V4 [34].

Для глобальних/локальних ознак будемо використовувати навчену модель у роботі DELG [8]. Параметри геометричної верифікації залишимо як у попередньому розділі. Серед гіперпараметрів розглянемо такі параметри:

- Поріг детекції (будемо розглядати тільки об’єкти з оцінкою більше ніж цей поріг).
- Негативний поріг (для негативних семантичних класів для верифікації).
- Негативна пропорція (поріг пропорції негативних детекцій).

Порівняння гіперпараметрів семантичної верифікації викладено в табл. 3. Можна побачити, що наявність семантичної верифікації значно покращує результати пошуку схожих зображень. Також можна помітити, що при одних і тих самих параметрах результати при використанні Mobile Net + SSD та Inception-Resnet V4 недуже відрізняються.

Таблиця 3 – Порівняння гіперпараметрів семантичної верифікації

Модель	Поріг дет.	Нег. поріг	Нег. пропорція	Валідація ЗСТ	Тестування ЗСТ
Без верифікації	-	-	-	0.4842	0.4575
MobileNet + SSD	0.0	0.3	0.6	0.4863	0.4624
	0.0	0.15	0.6	0.4899	0.4636
	0.1	0.15	0.6	0.4808	0.4576
	0.0	0.1	0.6	0.4918	0.4632
	0.0	0.07	0.5	**0.4955**	**0.4681**
Inception-Resnet-V4 + FRCNN	0.0	0.15	0.8	0.4919	0.4652
	0.0	0.15	0.7	0.4934	0.4678
	0.0	0.15	0.6	0.4940	0.4678

5.4. Результати навчання

Зробимо вивчення абляції натренованих моделей ResNet 152 V2 та Efficient Net B6 – спочатку з такими ж самими параметрами, що описані у статті DELG [8], а потім з власними покращеннями та модифікаціями. Розглянемо такі параметри:

- Архітектура моделі.
- Розміри або коефіцієнти масштабування вхідної піраміди зображень.
- Метод навчання.

Для порівняння також наведемо результати опублікованої моделі [30]. Для геометричної верифікації використаємо такі фіксовані параметри:

- Кількість ітерації RANSAC: 100’000.
- Максимальна кількість невикидів: 20.
- Максимальна похибка репроекції: 4.
- Довірчий поріг проективного перетворення: 0.99

Результати в табл. 4 наведені без використання семантичної верифікації.

Таблиця 4 – Результати навчених мереж без семантичної верифікації

Архітектура	Вхідна піраміда	Методи навчання	Тестування ЗСТ	Валідація ЗСТ
ResNet 101 [30]	(1/√2, 1, √2)	DELG [8]	0.4824	0.4578
ResNet 152 V2	512x512	DELG [8]	0.4901	0.5063
	(1/√2, 1, √2)		0.5030	0.5155
	(1, √2, 2)		0.5057	0.5225
	(1/√2, 1, √2)	Sub-center ArcFace + Dynamic Margin + Data Echoing	0.5099	0.5284
	(1, √2, 2)		0.5117	0.5304
Eff. Net B6	(1/√2, 1, √2)		0.5028	**0.5389**
	(1, √2, 2)		**0.5165**	0.5337
Eff. Net B5	(1/√2, 1, √2)		**0.5029**	0.5202

Можна побачити, що результати навчання із запропонованим методом ArcFace з підцентрами та мінливими відступами значно кращі, ніж простий ArcFace, що було використано в [8].

5.4.1. Ансамбль декількох нейронних мереж та семантична верифікація

Можна ще більше покращити результати, що наведені в попередньому розділі, за допомогою ансамблю декількох моделей. У таблиці нижче наведені результати ансамблів. Кожна з нейронних мереж ансамблів була натренована методом ArcFace з підцентрами та мінливими відступами. Також були зафіксовані наступні параметри геометричної верифікації:

- Кількість ітерації RANSAC: 100’000.
- Максимальна кількість невикидів: 20.
- Максимальна похибка репроекції: 4.
- Довірчий поріг проективного перетворення: 0.99.

Також зафіксовані параметри семантичної верифікації:

- Поріг детекції: 0.0.
- Негативний поріг: 0.07.
- Негативна пропорція: 0.5.

У зв’язку з обмеженнями по часу (9 годин на тестування на валідаційному та тестовому наборі даних) замість піраміди вхідних зображень (результати яких наведені у таблиці в попередньому розділі) було використано лише оригінальний розмір зображення.

Вибір коефіцієнтів для комбінування ознак різних нейронних мереж в ансамблі є також дуже важливою частиною побудови ансамблю. Були обрані ваги 1.0, 0.8, та 0.6 для нейронних мереж Efficient Net B6, ResNet 152 V2 та Efficient Net B5 відповідно.

Таблиця 5 – Кінцевий ансамбль навчених нейронних мереж разом із семантичною та геометричною верифікаціями

Ансамбль	Семантична верифікація	Валідація ЗСТ	Тестування ЗСТ
ResNet 152 V2 + Efficient Net B6	MobileNet SSD	0.5199	0.5368
ResNet 152 V2 + Efficient Net B6	Inception-Resnet V4 + F-RCNN	0.5308	0.5625
RN 152 V2 + EffNet B5 + EffNet B6	Без верифікації	0.5323	0.5642
ResNet 152 V2 + Efficient Net B6	Inception-Resnet V4 + F-RCNN	**0.5351**	**0.5642**

Отже, використання ансамблю значно може покращити кінцеві результати пошуку схожих зображень у базі.

5.5. Приклади

Нижче наведено приклади повернених базою глобальних ознак зображень та візуалізація знайдених локальних ознак.

На прикладах церкви, храму та Сіднейського оперного театру можна бачити, що локальні ознаки були гарно виділені та до них було знайдено відповідну пару в майже всіх випадках (рис. 28–33).

Рисунок 28 – Результати пошуку схожих зображень, церква

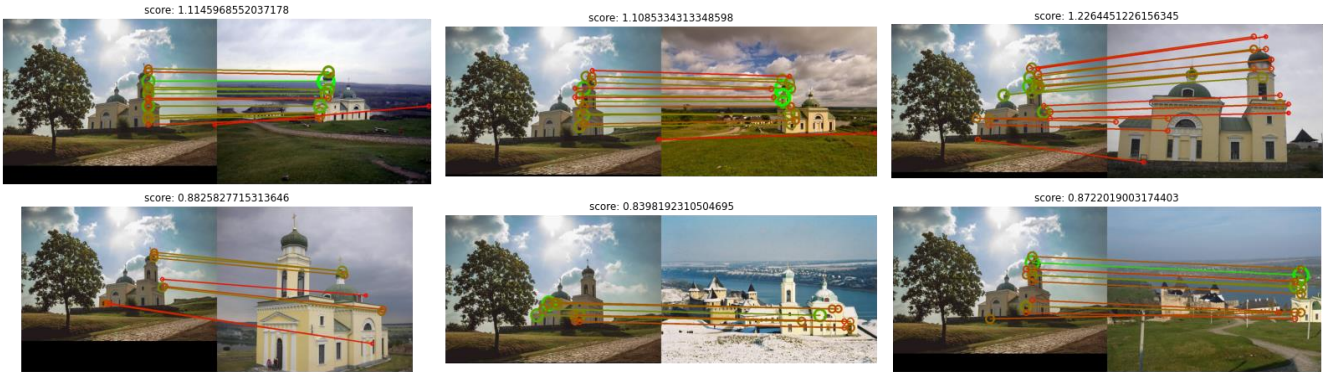

Рисунок 29 – Відповідність між локальними ознаками, церква

Рисунок 30 – Результати пошуку схожих зображень, Сіднейський оперний театр

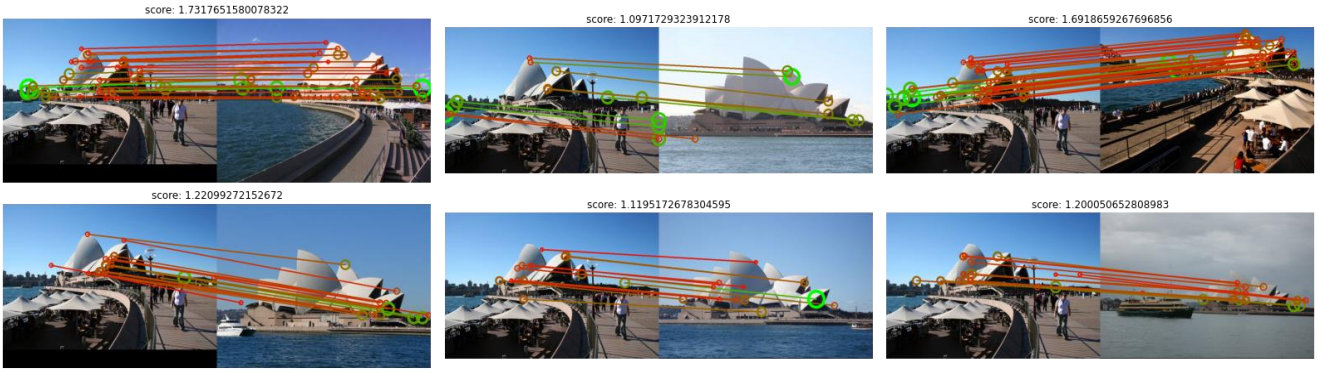

Рисунок 31 – Відповідність між локальними ознаками, Сіднейський оперний театр

Рисунок 32 – Результати пошуку схожих зображень, храм

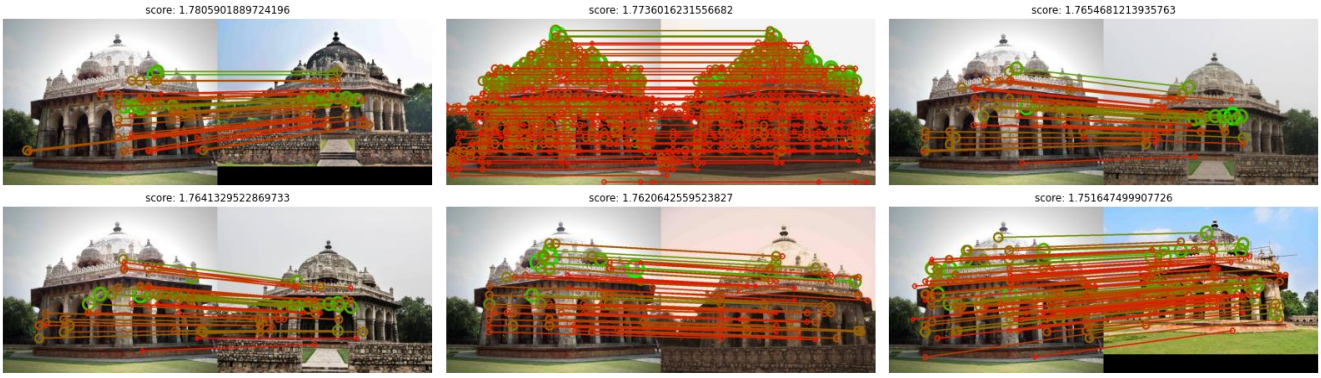

Рисунок 33 – Відповідність між локальними ознаками, храм

На прикладах мосту (рис. 34) та Ейфелевої вежі (рис. 36) можна побачити, що для деяких об’єктів кількість виділених локальних ознак не є великою, і не завжди до них коректно знаходиться відповідна пара. Так, у випадку моста деякі локальні ознаки виділені на небі, яке не є частиною об’єкта, що розглядається (рис. 35), а на Ейфелевій вежі забагато схожих локальних ознак, тому можна побачити, що деякі пари локальних ознак було зіставлено некоректно (рис. 37).

Рисунок 34 – Результати пошуку схожих зображень, міст

Рисунок 35 – Відповідність між локальними ознаками, міст

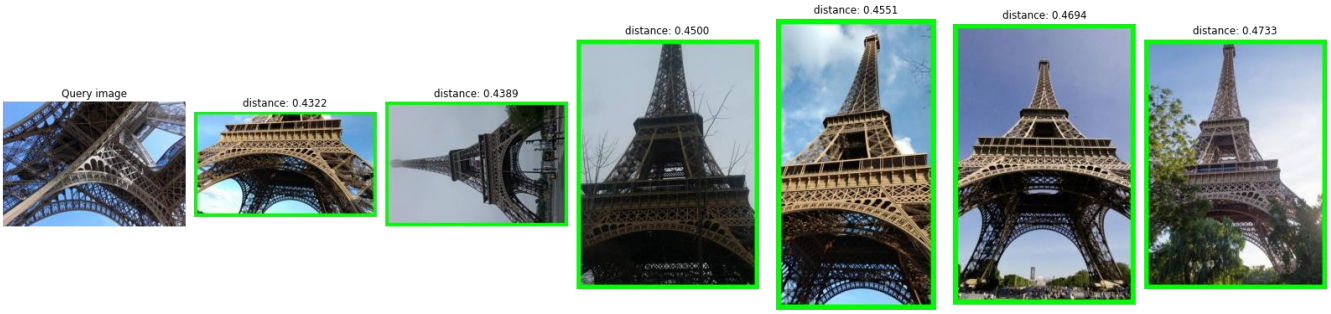

Рисунок 36 – Результати пошуку схожих зображень, Ейфелева вежа

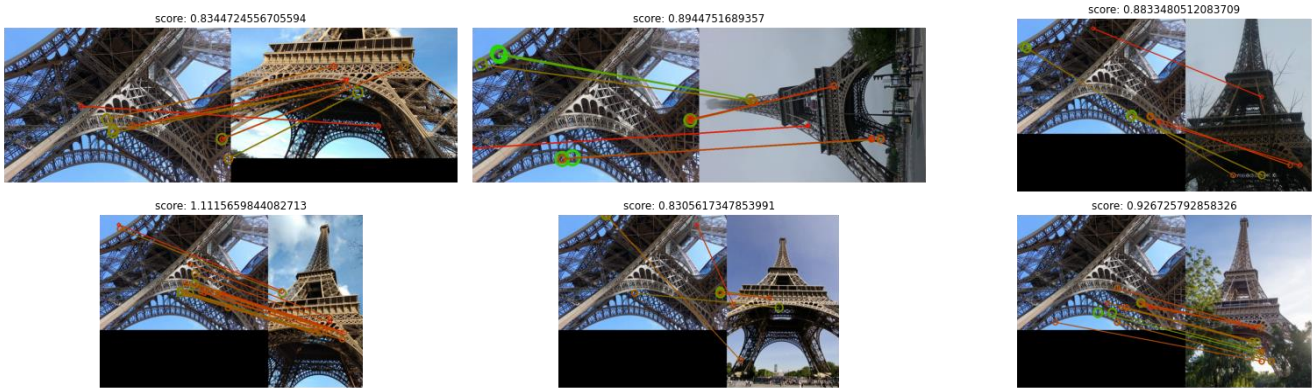

Рисунок 37 – Відповідність між локальними ознаками, Ейфелева вежа

ВИСНОВКИ

Метою цієї роботи була розробка системи на основі глибокої нейронної мережі та її навчання для знаходження зображень, схожих на задане, у масивній базі даних. Для досягнення цієї мети було успішно зроблено такі кроки.

- Запропоновано покращення функції втрат для покращення виразності та дискримінативності глобальних ознак порівняно з моделями DELF [9] та DELG [8], що є найкращими на цей час. Розроблений метод ArcFace з підцентрами та мінливими відступами є більш доречним, ніж звичайний ArcFace для метричного навчання в загальному випадку.
- Запропоновано покращення процесу навчання за допомогою використання циклічного планування кроку та мінливому збільшенню відступу для функції втрат у кінці кожного циклу.
- Запропоновано покращення швидкості навчання та утилізацію обчислювальних ресурсів за допомогою відлуння даних.
- Запропоновано покращення етапу ранжування результатів за допомогою локальних ознак, використовуючи семантичну верифікацію та покращення гіперпараметрів геометричної верифікації.

Загальна середня точність результатів пошуку схожих зображень за допомогою натренованої моделі є вищою за точність DELF [9] та DELG [8], а за швидкістю розроблена модель не є гіршою. Семантична верифікація та додаткова нейронна мережа були додані за рахунок звільненого обчислювального бюджету через відсутність подачі піраміди зображень на вхід. Однак, розроблена мережа працює так само добре з пірамідою на вході як і без неї, що додатково говорить про ефективність та якість натренованих глобальних ознак порівняно з останніми на цей час опублікованими методами.

Розроблена мережа була натренована на наборі даних локацій, і може використовуватися для пошуку схожих зображень у, наприклад, пошукових системах чи галереях зображень (альбомах фотографій), коли користувач не

може визначити об’єкт пошуку в текстовому вигляді, а також для визначення поточного місцезнаходження роботами.

Розроблена система розподіленого навчання є доволі узагальненою і може бути використана в інших проектах великомасштабного комп’ютерного зору, обробки природної мови. Усі проведені експерименти можуть бути повторені заново з тим самим результатом, що говорить про стабільність розробленого методу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lowe D. G. Distinctive Image Features from Scale-Invariant Keypoints / David G. Lowe // International Journal of Computer Vision. – 2004. – Vol. 60. – No. 2 – P. 91–110.
2. Bay H. SURF: Speeded Up Robust Features / Herbert Bay, Tinne Tuytelaars, Luc Van Gool // Computer Vision – ECCV 2006: Lecture Notes in Computer Science / A. Leonardis, H. Bischo, A. Pinz (eds.). – Springer; Berlin; Heidelberg, 2006. – Vol. 3951. – P. 404-417.
3. Sivic J. Video Google: a text retrieval approach to object matching in videos / Josef Sivic, Andrew Zisserman // Proceedings Ninth IEEE International Conference on Computer Vision. – Nice, 2003. – Vol. 2. – P. 1470–1477.
4. Fischler M. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography / Martin A. Fischler, Robert C. Bolles // Communications of the ACM : journal. – 1981. – Vol. 24. – P. 381–395.
5. Krizhevsky A. ImageNet Classification with Deep Convolutional Neural Networks / Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton // NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems. – Curran Associated Inc., Red Hook, 2012. – Vol. 1. – P. 1097–1105.
6. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis / Hajime Taira, Masatoshi Okutomi, Torsten Sattler et al. // 2018 IEEE/CVF. Conference on Computer Vision and Pattern Recognition. – Salt Lake City, 2018. – P. 7199–7209.
7. NetVLAD: CNN architecture for weakly supervised place recognition / Relja Arandjelovic´, Petr Gronat, Akihiko Torii et al. // 2016 IEEE. Conference on Computer Vision and Pattern Recognition (CVPR). – Las Vegas, 2016. – P. 5297–5307.

8. Cao B. Unifying Deep Local and Global Features for Image Search / Bingyi Cao, André Araujo, Jack Sim // Proceedings of the European Conference on Computer Vision (ECCV). – Glasgow, 2020. – P. 726–743.
9. Large-Scale Image Retrieval with Attentive Deep Local Features / Hyeonwoo Noh, André Araujo, Jack Sim et al. // 2017 IEEE. International Conference on Computer Vision (ICCV). – Venice, 2017. – P. 3476–3485.
10. Google Landmarks Dataset v2 – A Large-Scale Benchmark for Instance-Level Recognition and Retrieval / Tobias Weyand, André Araujo, Bingyi Cao et al. // 2020 IEEE/CVF. Conference on Computer Vision and Pattern Recognition (CVPR). – On-line, 2020. – P. 2572–2581.
11. Kaya M. Deep Metric Learning: A Survey / Mahmut Kaya, Hasan Şakir Bilge // Symmetry. 2019. – Vol.11. – No.9. – P. *1066.*
12. Deep Residual Learning for Image Recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren et al. // 2016 IEEE. Conference on Computer Vision and Pattern Recognition (CVPR). – Las Vegas, 2016. – P. 770–778.
13. Identity Mappings in Deep Residual Networks / Kaiming He, Xiangyu Zhang, Shaoqing Ren et al. // Computer Vision – ECCV 2016. 14th European Conference on Computer Vision (ECCV 2016) : Lecture Notes in Computer Science / B. Leibe, J. Matas, N. Sebe, M. Welling (eds.). – Amsterdam, 2016. – Vol. 9908. – P. 630–645.
14. Tan M. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks / Mingxing Tan, Quoc V. Le // 36th International Conference on Machine Learning (ICML 2019). – Long Beach, 2019. – P. 6105-6114.
15. ImageNet: A large-scale hierarchical image database / Jia Deng, Wei Dong, Richard Socher et al. // 2009 IEEE. Conference on Computer Vision and Pattern Recognition. – Miami, 2009. – P. 248–255.
16. MnasNet: Platform-Aware Neural Architecture Search for Mobile / Mingxing Tan, Bo Chen, Ruoming Pang et al. // 2019 IEEE/CVF. Conference on Computer Vision and Pattern Recognition (CVPR). – Long Beach, 2019. – P. 2815–2823.

17. ArcFace: Additive Angular Margin Loss for Deep Face Recognition / Jiankang Deng, Jia Guo, Niannan Xue et al. // 2019 IEEE/CVF. Conference on Computer Vision and Pattern Recognition (CVPR). – Long Beach, 2019. – P. 4685–4694.
18. Sub-center ArcFace: Boosting Face Recognition by Large-Scale Noisy Web Faces / Jiankang Deng, Jia Guo, Tongliang Liu et al. // Computer Vision – ECCV 2020. ECCV 2020: Lecture Notes in Computer Science / A. Vedaldi, H. Bischof, T. Brox, J. Frahm (eds.). – Glasgow, 2020. – Vol. 12356. – P. 741–757.
19. Siu C. Residual Networks Behave Like Boosting Algorithms / Chapman Siu // 2019 IEEE. International Conference on Data Science and Advanced Analytics (DSAA). – Washington, 2019. – P. 31–40.
20. Chum O. Two-view geometry estimation unaffected by a dominant plane / O. Chum, Tomás Werner, Jiri Matas // 2005 IEEE. Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). – San Diego, 2005. – Vol. 1. – P. 772–779.
21. Bentley J. Multidimensional Binary Search Trees Used for Associative Searching / Jon Louis Bentley // Communications of the ACM: journal. – 1975. – Vol. 18. – P. 509–517.
22. Kingma D. Adam: A Method for Stochastic Optimization / Diederik P. Kingma, Jimmy Lei Ba // 3rd International Conference *for* Learning Representations. – San Diego, 2015.
23. Lv K. Learning Gradient Descent: Better Generalization and Longer Horizons / Kaifeng Lv, Shunhua Jiang, Jian Li // Proceedings of the 34th International Conference on Machine Learning. – Sydney, 2017. – P. 2247–2255.
24. Loshchilov I. SGDR: Stochastic Gradient Descent with Warm Restarts / Ilya Loshchilov, Frank Hutter // 5th International Conference on Learning Representations (ICLR 2017) Conference Track. – Toulon, 2017.
25. Snapshot Ensembles: Train 1, get M for free / Gao Huang, Yixuan Li, Geoff Pleiss et al. // 5th International Conference on Learning Representations (ICLR 2017) Conference Track. – Toulon, 2017.

26. AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations / Xiao Zhang, Rui Zhao, Yu Qiao et al. // 2019 IEEE/CVF. Conference on Computer Vision and Pattern Recognition (CVPR). – Long Beach, 2019. – P. 10815–10824.
27. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features / Sangdoo Yun, Dongyoon Han, Seong Joon Oh et al. // 2019 IEEE/CVF. International Conference on Computer Vision (ICCV). – Long Beach, 2019. – P. 6022–6031.
28. Faster Neural Network Training with Data Echoing / Dami Choi, Alexandre Passos, Christopher J. Shallue et al. // International Conference on Learning Representations (ICLR 2019). – New Orleans, 2019.
29. Perronnin F. A family of contextual measures of similarity between distributions with application to image retrieval / Florent Perronnin, Yan Liu, Jean-Michel Renders // IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009). – Miami, 2009. – P. 2358–2365.
30. DELG models. 2020 [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/tensorflow/models/tree/master/research/delf
31. Hartley R. Multiple View Geometry in computer vision / Richard Hartley, Andrew Zisserman. – Second edition. – Cambridge University Press, 2003. – 672 p.
32. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications / Andrew G. Howard, Menglong Zhu, Bo Chen et al. // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – Honolulu, 2017.
33. SSD: Single Shot MultiBox Detector / Wei Liu, Dragomir Anguelov, Dumitru Erhan et al. // Computer Vision – ECCV 2016. 14th European Conference on Computer Vision (ECCV 2016) : Lecture Notes in Computer Science / B. Leibe, J. Matas, N. Sebe, M. Welling (eds.). – Amsterdam, 2016. – Vol. 9905. – P. 21–37.
34. Open Images Dataset V6 + Extensions. 2020 [Електронний ресурс]. – Режим доступу до ресурсу: https://storage.googleapis.com/openimages/web/index.html

35. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning / Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke et al. // AAAI Publications, Thirty-First AAAI Conference on Artificial Intelligence. – San Francisco, 2016. – P. 4278–4283.
36. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / Shaoqing Ren, Kaiming He, Ross Girshick et al. // NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems. – MIT Press, Cambridge, 2015. – Vol. 1. – P. 91–99.

ДОДАТОК А

Програмний пакет

Код на мові python з використанням фреймворку TensorFlow етапів навчання та тестування розробленої мережі, що представлена в цій роботі, розміщено за посиланням: https://bitbucket.org/a_andreichuk/masterthesis/