КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття осівтнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА ІНТЕЛЕКТУАЛЬНОГО ДОДАТКУ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

Виконав студент 4-го курсу
Аркадій ЦИГАНОВ _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Лариса КАТЕРИНИЧ _____
(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 інтелектуальних програмних систем
« __ » _____ 2023 р., протокол № __
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 43 сторінок, 17 ілюстрацій, 16 джерел посилань.

IMAGE CLASSIFICATION, TRANSFER LEARNING, NEAREST NEIGHBOR ALGORITHM, IMAGE RETRIEVAL TECHNIQUES, IMAGE REPRESENTATIONS, TRIPLET LOSS, PYTORCH, OPEN CV, SHOOL LUNCH DATASET.

Об'єкт дослідження: дослідження розробки системи класифікації зображень їжі на тацях за допомогою використання представлення зображення у вигляді ембеддингів і пошуку найближчого сусіда. Для розрахунку швидкої ціни їжі на таці.

Мета роботи: ознайомлення зі сферою розробки систем класифікації зображень за допомогою використання вектору ембеддингів, розробка прототипу системи класифікації зображень їжі на таця.

Методи та інструменти розробки: мова програмування Python, бібліотека PyTorch для реалізації моделі класифікації зображень, бібліотека OpenCV для обробки та аналізу зображень, використання техніки triplet loss для навчання моделі на векторах ембеддингів, середовище розробки Microsoft Visual Studio.

Результати роботи: був проведений аналіз сучасних методів класифікації зображень та технологій для розробки систем класифікації. На основі цього аналізу була розроблена система класифікації зображень їжі на тацях за допомогою представлення зображення у вигляді вектору ембеддингів і пошуку найближчого сусіда. Також було реалізовано функціонал для швидкого розрахунку ціни їжі на таці.

Сфера застосування: система класифікації зображень їжі на тацях може бути використана в ресторанах, кафе та інших закладах громадського харчування для автоматизації процесу розрахунку ціни їжі. Вона дозволяє визначати склад та кількість продуктів на таці, що сприяє покращенню обслуговування та уникнення помилок під час розрахунку ціни. І така система може бути корисною в сфері кейтерингу та організації заходів, де необхідно швидко оцінити вартість страв на основі зображень їжі на тацях.

Значимість роботи і пропозиції щодо розвитку: на даному етапі розробки система реалізовує основний функціонал класифікації зображень та розрахунку ціни. Однак, для подальшого розвитку можна розширити функціонал системи, додавши можливості автоматичного визначення складу та вартості інгредієнтів, а також створення бази даних з цінами на різні продукти. Таким чином, розробка системи класифікації зображень їжі на тацях за допомогою представлення зображення у вигляді вектору ембеддингів і пошуку найближчого сусіда є актуальною та перспективною задачею. Вона може сприяти автоматизації та оптимізації процесу розрахунку ціни їжі, що має значний потенціал у галузі громадського харчування та кейтерингу.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

CNN - Convolution neural network, згорткові нейронні мережі;

ML - machine learning, машинне навчання;

FC - fully connected layers, повністю пов'язані шари в нейронах мережах;

CPU - computer programmer unit, процесор;

CUDA - Compute Unified Device Architecture, тип графічних процесорів;

HSV - Hue Saturation Value;

VGG - Visual Geometry Group;

TL - transfer learning;

k-NN - k-nearest neighbors;

HTTP - Hypertext Transfer Protocol;

GUI- Graphical User Interface.

ВСТУП

Останнім часом спостерігається значне зростання кількості закладів харчування, що пропонують таку ідею оплати: клієнтам пропонується сплатити за замовлення, яке вони отримують на своїй таці. Цей підхід забезпечує швидкий сервіс і зменшує час очікування на касі.

Концепція цих закладів полягає у тому, що клієнт робить свій вибір з стенду з їжею, звідки він її кладе собі на тацю. Далі проходить на касу і касир вручну вбиває кожну позицію блюда в комп’ютер.

Однак, зі зростанням популярності цих закладів і збільшенням потоку клієнтів виникає нова проблема. Каси, призначені для обробки платежів, не завжди встигають швидко опрацювати кожного клієнта протягом короткого проміжку часу. Це призводить до утворення черги на касі і збільшує час очікування, що суперечить ідеї швидкого обслуговування, яку пропагують ці заклади.

Рішення цієї проблеми може бути в автоматизації розрахунку ціни.

Мета цієї роботи полягає в розробці прототипу системи, яка зможе аналізувати фотографії таці зверху, класифікувати кожне блюдо на таці. Ця система створюється з метою оптимізації процесу обслуговування в закладах харчування, де оплата здійснюється на основі набраної їжі на таці.

Для досягнення зазначеної вище мети необхідно виконати наступні завдання:

1. Знайти підходящий датасет з зображеннями їжі зверху на таці

2. Застосування попередньої обробки до зображень для отримання кожного блюда на окремій фотографії. Це може включати зменшення розміру зображень, видалення шуму або покращення контрастності.

3. Використання глибокого навчання, зокрема згорточних нейронних мереж (Convolutional Neural Networks, CNN) з transfer learning і отримування характеристик у вигляді вектору з зображення.

4. Використовуючи характеристик у вигляді вектору кожного зображення навчити глибокого нейрону межу проектувати у вектор ембедінгів у великовимірному просторі таким чином щоб ембедінги одного класу знаходились якомога ближче один до одного, а різних класів якнайдалі.

5. Класифікувати блюда за допомогою алгоритму ближчого сусіда і відстані між вектором ембедінгами зображень блюд

6. Реалізувати розроблену систему в програмному середовищі, яке зможе обробляти фотографії, виконувати класифікацію страв.

7. Провести тестування системи з різними зображеннями страв для оцінки її точності та продуктивності.

РОЗДІЛ 1 ЗАГАЛЬНІ МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ І КЛАСИФІКАЦІЙ

1.1 Згорточна нейронни мережі (CNN)

Згорточна нейронна мережа (CNN) — алгоритм глибокого навчання, який дозволяє аналізувати вхідні зображення і приділяти вагомість (вагові коефіцієнти та зміщення) різним аспектам та об'єктам на зображенні, а також розрізняти їх один від одного. У порівнянні з іншими алгоритмами класифікації, попередня обробка, необхідна для CNN, є значно меншою. Замість ручної настройки фільтрів, як це робиться в примітивних методах, CNN може вивчати ці фільтри та характеристики після достатнього навчання.

Архітектура CNN схожа на підключення нейронів у людському мозку. Окремі нейрони реагують на стимули лише в обмеження області поля зору, яка називається рецептивним полем. Набір таких полів перекривається, щоб охопити всю візуальну область. Зазвичай на фотографіях ми маємо зображення у форматі RGB, що складається з трьох кольорових площин - червоної, зеленої та синьої. Існує кілька кольорових просторів, в яких можуть бути представлені зображення, наприклад, відтінки HSV, CMYK та інші.

Рух фільтра - Фільтр переміщається вправо з певним значенням кроку, доки не проаналізує всю ширину. Рухаючись далі, він стрибає вниз до початку (ліворуч) зображення з тим самим значенням кроку та повторює процес, доки не буде пройдено все зображення. (Рисунок 1.1)

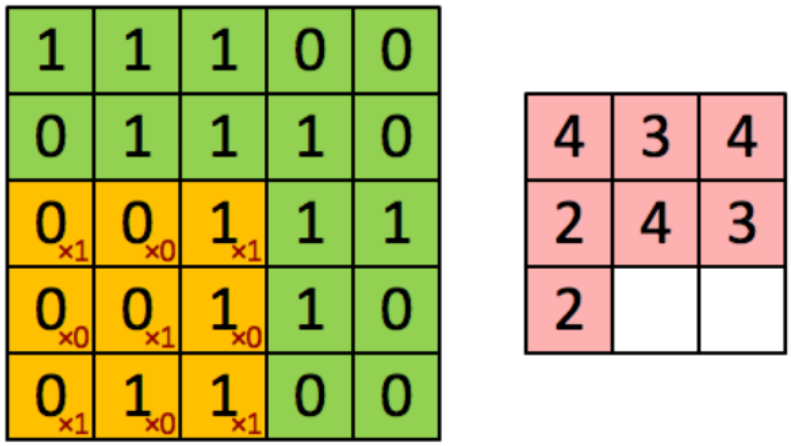

Рисунок 1.1 Приклад роботи фільтра в CNN

Зліва зображення 5x5 і фільтр 3x3. Справа Проміжний результат роботи фільтра.

Мета операції згортання полягає в тому, щоб витягти високорівневі характеристики, такі як ребра, із вхідного зображення. CNN не повинні обмежуватися лише одним згортковим рівнем. Традиційно перший CNN відповідає за захоплення функцій низького рівня, таких як краї, колір, орієнтація градієнта тощо. З доданими шарами архітектура також адаптується до функцій високого рівня, надаючи нам мережу з повним розумінням зображень у наборі даних, як і ми.

Існує два типи результатів операції — один, у якому розмірність згорнутого об’єкта зменшується порівняно з вхідними, а інший — у якому розмірність або збільшується, або залишається незмінною. Це робиться шляхом застосування дійсного відступу у випадку першого або однакового відступу у випадку останнього.

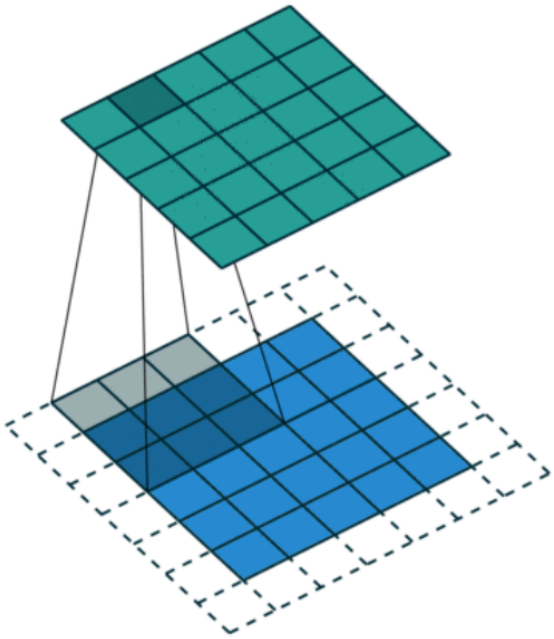

Рисунок 1.2 Згортка, коли розмірність результату залишається незмінною

Коли ми збільшуємо зображення 5x5x1 до зображення 6x6x1, а потім застосовуємо ядро 3x3x1 до нього, ми виявляємо, що згорнута матриця має розміри 5x5x1. Звідси і назва — Same Padding (Рисунок 1.2).

З іншого боку, якщо ми виконаємо ту саму операцію без доповнення, ми отримаємо матрицю, яка має розміри самого ядра (3x3x1) — допустиме доповнення.

1.2 Рівень об’єднання (Pooling Layer)

Подібно до згорткового рівня, рівень об’єднання відповідає за зменшення просторового розміру згорнутого об’єкта. Це робиться для зменшення обчислювальної потужності, необхідної для обробки даних, шляхом зменшення розмірності. Крім того, це корисно для виділення домінуючих характеристик, які є обертальними та позиційними інваріантними, таким чином підтримуючи процес ефективного навчання моделі.

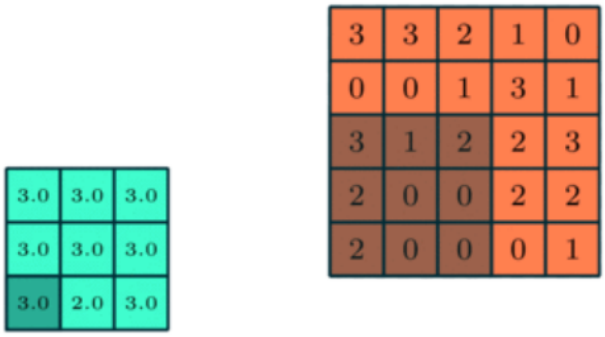

Рисунок 1.3 Приклад роботи Max Pooling

Існують два типи операцій згорткового пулінгу: Max Pooling (Рисунок 1.3) та Average Pooling. Max Pooling повертає найбільше значення з певної області зображення, охопленої ядром. З іншого боку, Average Pooling повертає середнє значення всіх пікселів в тій же області. (Рисунок 1.4).

Max Pooling може використовуватися як засіб позбавлення від шумів. Він повністю відкидає шумові активації та сприяє зменшенню розмірності зображення. З іншого боку, Average Pooling просто зменшує розмірність як механізм підганяння шумів. Таким чином, можна сказати, що Max Pooling працює більш ефективно, ніж Average Pooling.

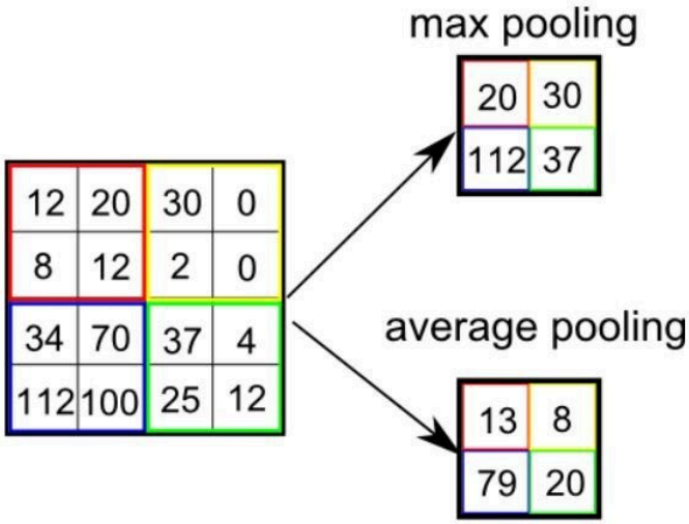

Рисунок 1.4 Різниця між результатами Max Pooling і Average Pooling

Згортковий рівень і рівень Pooling разом утворюють і-й рівень згорткової нейронної мережі. Залежно від складності зображень, кількість таких шарів може бути збільшена для ще більшого захоплення деталей низького рівня, але ціною більшої обчислювальної потужності.

1.3 Опис VGG16

VGG16 є одним з найкращих типів згорткових нейронних мереж (CNN) в галузі комп'ютерного зору на сьогоднішній день. Його творці підвищили глибину мережі, використовуючи архітектуру зі згортковими фільтрами розміром 3 × 3, що значно покращило ефективність порівняно з попередніми методиками. Вони розширили глибину до 16-19 шарів вагових параметрів.

VGG16 є алгоритмом виявлення та класифікації об'єктів, здатним класифікувати 1000 зображень у 1000 різних категорій з точністю 92,7%. Це один з популярних алгоритмів класифікації зображень, який легко використовувати з використанням передварительно навчених моделей. У назві VGG16 число 16 відноситься до кількості шарів з вагами. У VGG16 є 13 згорткових шарів, 5 шарів

пулінгу та 3 повнозв'язаних шари, загалом 21 шар, проте фактично він має тільки 16 шарів з вагами, тобто кількість параметрів, які можуть бути навчені.

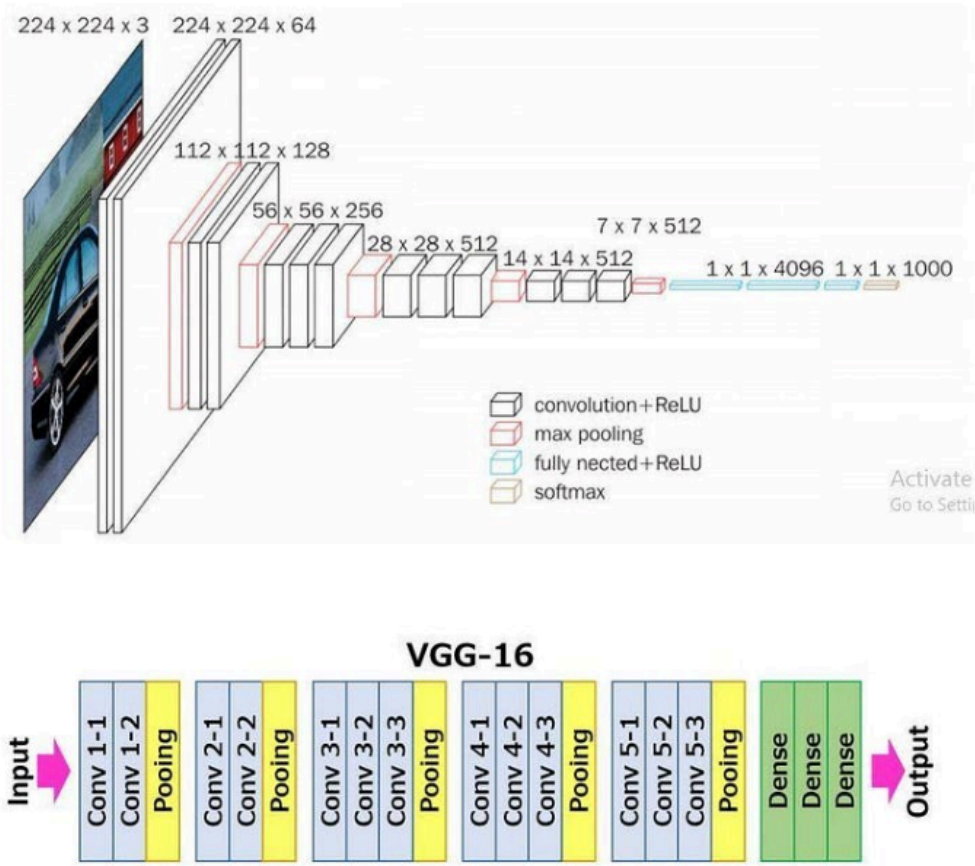

Рисунок 1.5 Архітектура VGG16

Розмір вхідного тензора для VGG16 приймається як 224 на 224 з 3 каналами RGB (Рисунок 1.5). Однією з найвизначальніших особливостей VGG16 є застосування багатьох шарів згортки з фільтрами розміром 3x3 з кроком 1 та постійним рівнем заповнення, а також використання шарів максимального пулінгу з фільтрами розміром 2x2 та кроком 2. Шари згортки та Max Pool послідовно розташовані по всій архітектурі.

Рівень Conv-1 містить 64 фільтри, Conv-2 має 128 фільтрів, Conv-3 має 256 фільтрів, Conv 4 і Conv 5 має 512 фільтрів.

Три повністю з’єднані (FC) шари слідують за стеком згорткових шарів: перші два мають по 4096 каналів кожен, третій виконує 1000-сторонню класифікацію ILSVRC і, таким чином, містить 1000 каналів (по одному для кожного класу). Останнім шаром є шар soft-max.

1.4 Трансферне навчання (TL)

Трансферне навчання (TL) - це метод машинного навчання (ML), який зосереджений на використанні знань, отриманих під час розв'язання одного завдання, для вирішення пов'язаного завдання. Наприклад, знання, набуті під час навчання моделі розпізнавання автомобілів, можуть бути використані для розпізнавання вантажівок. Ця концепція має відношення до психологічних досліджень щодо перенесення навичок, хоча практичні зв'язки між цими двома галузями обмежені.

Використання інформації, отриманої з раніше вивчених завдань, в нових завданнях має потенціал для суттєвого покращення ефективності навчання.

Трансферне навчання передбачає перенесення знань з однієї навченої моделі машинного навчання на іншу, але схожу задачу. Наприклад, якщо ви навчили простий класифікатор розпізнавати, чи присутній рюкзак на зображенні, ви можете використати знання цієї моделі для ідентифікації інших об'єктів, наприклад, сонцезахисних окулярів.

За допомогою трансферного навчання ми намагаємося в основному використовувати вивчені знання з одного завдання, щоб краще зрозуміти концепції іншого завдання. Ваги моделі автоматично переносяться на мережу, яка виконує "завдання A" замість мережі, яка раніше вирішувала "завдання B".

Через величезну потужність CPU, необхідну для передачі навчання, зазвичай застосовують комп’ютерне бачення та завдання обробки природної мови.

У комп’ютерному зорі нейронні мережі зазвичай спрямовані на виявлення країв у першому шарі, форм у середньому шарі та специфічних для завдання функцій у останніх шарах. Початковий і центральний рівні використовуються для

перехідного навчання, а останні рівні лише перенавчаються. Він використовує позначені дані із завдання, якому він навчався. (Рисунок 1.6)

Оскільки модель уже була попередньо навчена, хорошу модель машинного навчання можна створити з досить невеликою кількістю навчальних даних за допомогою трансферного навчання. Це особливо корисно при обробці природної мови, де величезні мічені набори даних вимагають багато експертних знань. Крім того, час навчання скорочується, оскільки створення глибокої нейронної мережі з самого початку складного завдання може зайняти дні або навіть тижні.

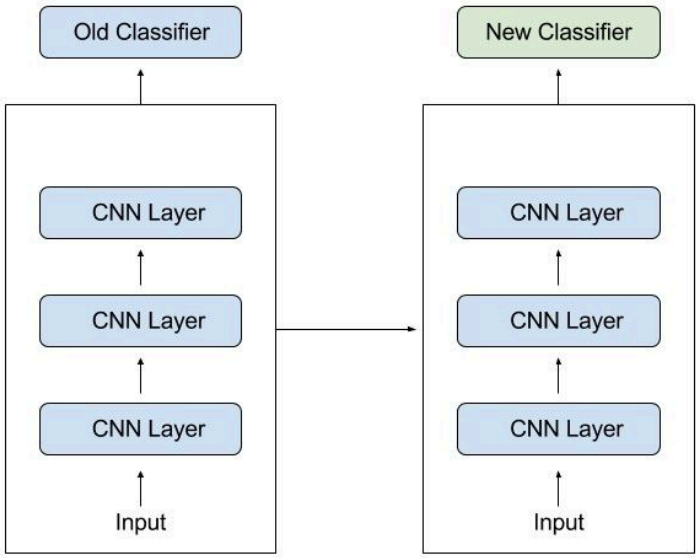

Рисунок 1.6 Трансферне навчання на новому класифікаторі

Нейронні мережі володіють здатністю визначати, які функції є критичними, а які - ні. Навіть для складних завдань, які раніше вимагали значних зусиль людини, алгоритм навчання може швидко знайти оптимальну комбінацію характеристик. Отримане в результаті навчання представлення може бути використане для різних інших завдань. Просто використовуйте початкові шари для знаходження відповідного представлення функцій, але уникайте використання вихідних даних мережі, оскільки вони сильно залежать від конкретного завдання. Замість цього подайте дані до вашої мережі та виймайте їх через один з проміжних шарів. Необроблені дані можна трактувати як представлення цього

шару. Цей метод часто використовується у комп'ютерному зорі, оскільки він дозволяє зменшити об'єм даних, скоротити час обчислень і зробити їх більш сумісними з класичними алгоритмами.

1.5 Triplet Loss

Triplet Loss (1) — це одна з найпопулярніших функцій втрати для контрольованої подібності або метричного вивчення. У своєму найпростішому поясненні Triplet Loss заохочують, щоб різнорідні пари були віддалені від будь-яких подібних пар принаймні на певне значення запасу. (Рисунок 1.7)

Математичне представлення Triplet Loss:

$$L = max(d(a,p) - d(a,n) + m,0)\ , \qquad (1)$$

де a - референсна точка

p - позинивна точка, така, яка має такий же клас як і a

n - негативна точка, має інший клас за a

d - функція відстані, буду використовувати евклідову відстань

m - це значення запасу, щоб негативні зразки залишалися далеко один від одного

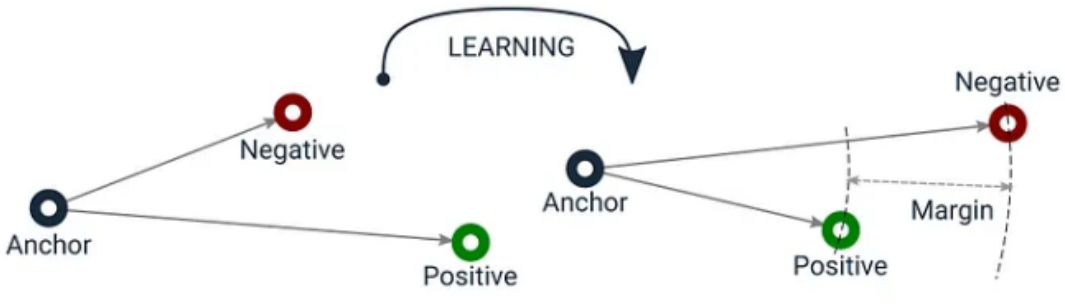

Рисунок 1.7 Приклад успішного застосування Triplet Loss

Слід зазначити, що Triplet Loss не має такого побічного ефекту, як у випадку Contrastive Loss, де опорні та позитивні зразки спонукаються збігатися в одну

точку векторного простору. Triplet Loss дозволяє певну дисперсію всередині класу, навпаки від Contrastive Loss, який вимагає, щоб відстань між якорем та будь-яким позитивним зразком була точно рівна нулю. Іншими словами, Triplet Loss дозволяє розтягувати кластери таким чином, що вони можуть включати викиди, при цьому забезпечуючи достатній проміжок між зразками з різних кластерів, наприклад, негативними парами.

Крім того, Triplet Loss менш жадібний (Рисунок 1.8). У відмінність від Contrastive Loss, він задовольняється, коли різні зразки легко відрізнити від подібних. Він не вносить зміни в позитивний кластер, якщо немає перешкод від негативних прикладів. Це пояснюється тим, що Triplet Loss намагається забезпечити достатній проміжок між відстанями негативних пар і відстанями позитивних пар. З іншого боку, Contrastive Loss враховує значення межі тільки при порівнянні різнорідних пар, і йому цілком байдуже, де знаходяться подібні пари у цей момент. Це означає, що контрастна втрата може досягти локального мінімуму швидше, в той час як Triplet Loss може продовжувати поліпшувати векторний простір.

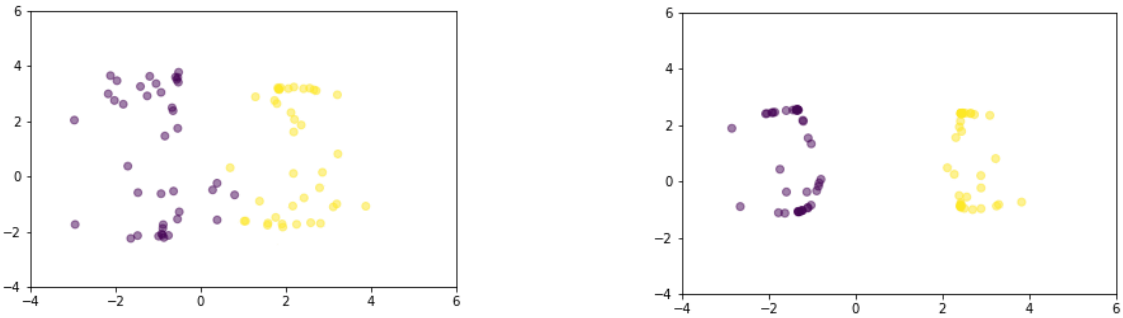

Рисунок 1.8 Демонстрація роботи Triplet Loss

Формулювання Triplet Loss демонструє, що він працює на трьох об’єктах одночасно:

1. Якір (референс)
2. Позитивний - зразок, який має ту ж мітку, що і якір
3. Негативний - зразок з відмінною від якоря і позитиву міткою.

1.6 Метод *k*-найближчих сусідів

Метод k-найближчих сусідів є непараметричним контрольованим методом навчання, розробленим у 1951 році Евелін Фікс та Джозефом Ходжесом, а пізніше розширеним Томасом Ковером. Цей метод використовується для класифікації та регресії. У обох випадках вхідні дані складаються з k найближчих навчальних прикладів з набору даних. Результат залежить від того, чи використовується k-NN для класифікації чи регресії:

- У класифікації за допомогою k-NN визначається клас. Об'єкт класифікується на основі голосування його сусідів, причому об'єкт призначається до класу, який є найпоширенішим серед k його найближчих сусідів (де k - ціле число, зазвичай невелике). Коли k = 1, об'єкт просто призначається до класу цього єдиного найближчого сусіда.

- У регресії за допомогою k-NN визначається значення властивості для об'єкта. Це значення обчислюється як середнє значень властивостей k найближчих сусідів. Коли k = 1, вихід просто стає рівним значенню цього єдиного найближчого сусіда.

k-NN є типом класифікації, де функція апроксимується локально, а всі обчислення відкладаються до оцінки функції. Оскільки цей алгоритм ґрунтується на відстані для класифікації, нормалізація навчальних даних може значно підвищити точність, особливо якщо об'єкти представляють різні фізичні одиниці або мають відмінні масштаби.

Корисним підходом як для класифікації, так і для регресії є використання вагових коефіцієнтів для врахування внеску сусідів, де ближчі сусіди мають більший вплив на середнє значення, ніж далекі сусіди. Наприклад, поширеною схемою вагового коефіцієнта є присвоєння кожному сусіду ваги 1/d, де d - відстань до сусіда.

Сусіди беруться з набору об’єктів, для яких відомий клас (для класифікації k-NN) або значення властивості об’єкта (для регресії k-NN). Це можна розглядати як навчальний набір для алгоритму, хоча чіткого етапу навчання не потрібно.

Особливістю алгоритму k-NN є те, що він чутливий до локальної структури даних.

1.7 Алгоритм методу *k*-найближчих сусідів

Навчальні приклади представлені у вигляді векторів у багатовимірному просторі ознак, і кожен з них має мітку класу. Фаза навчання алгоритму полягає в збереженні векторів ознак та міток класів навчальних зразків. Під час класифікації, k вважається заданою константою, і непозначений вектор (запит або тестова точка) класифікується шляхом присвоєння мітки, яка найчастіше зустрічається серед k найближчих навчальних зразків до цієї запитової точки. Зазвичай, для неперервних змінних використовується евклідова відстань як метрика відстані.

Для дискретних змінних, таких як класифікація тексту, можна використовувати іншу метрику, наприклад метрику перекриття (або відстань Хеммінга). У контексті даних мікрочипів експресії генів, наприклад, k-NN використовує коефіцієнти кореляції, такі як Пірсона та Спірмена, як метрику. Часто точність класифікації k-NN може значно покращитися, якщо метрика відстані вивчається за допомогою спеціалізованих алгоритмів, таких як аналіз найближчого сусіда з великим запасом або компонент сусідства.

Одним з недоліків базової класифікації "голосування за більшістю" є спотворення результатів, коли розподіл класів нерівномірний. Це означає, що приклади з більш поширеним класом, як правило, мають більший вплив на передбачення нового прикладу, оскільки вони частіше з'являються серед k найближчих сусідів через їх велику кількість.

Одним із способів подолання цієї проблеми є зважування класифікації, беручи до уваги відстань від тестової точки до кожного з її k найближчих сусідів. Клас (або значення, у задачах регресії) кожної з k найближчих точок множиться на вагу, пропорційну оберненій відстані від цієї точки до контрольної точки. Ще один спосіб подолати перекіс — це абстрагувати представлення даних.

Наприклад, у самоорганізовій карті кожен вузол є представником (центром) кластера подібних точок, незалежно від їхньої щільності у вихідних навчальних даних.

Вибір оптимального значення k залежить від конкретних даних. Зазвичай, збільшення значення k зменшує вплив шуму на класифікацію, але при цьому робить межі між класами менш чіткими. Існують різні евристичні методи для вибору відповідного значення k. Алгоритм, при якому клас прогнозується як клас найближчого навчального зразка (тобто k = 1), називається алгоритмом найближчого сусіда.

Точність алгоритму k-NN може серйозно знизитися внаслідок наявності зашумлених або нерелевантних ознак, або якщо значення ознак не відповідають їх важливості. Було проведено багато досліджень для вибору та масштабування ознак з метою покращення класифікації. Один з популярних підходів полягає у використанні еволюційних алгоритмів для оптимізації масштабування функцій. Інший підхід включає масштабування функцій на основі спільної інформації між навчальними даними та їх класами.

У випадках двокласової класифікації корисно обрати непарне значення k, оскільки це дозволяє уникнути рівних голосів. Один із популярних способів емпіричного вибору оптимального k - метод початкового завантаження.

Найбільш інтуїтивно зрозумілим класифікатором, що використовує найближчого сусіда, є класифікатор з одним найближчим сусідом, який присвоює точці x клас свого найближчого сусіда у просторі ознак.

Коли розмір навчального набору даних наближається до нескінченності, один найближчий сусідній класифікатор гарантує частоту помилок не гіршу, ніж подвоєна частота помилок Байєса (мінімальна досяжна частота помилок, враховуючи розподіл даних).

Наївну версію алгоритму легко реалізувати шляхом обчислення відстаней від тестового прикладу до всіх збережених прикладів, але вона потребує обчислень для великих навчальних наборів.

Використання алгоритму пошуку приблизного найближчого сусіда робить k-NN придатним для обчислень навіть для великих наборів даних.

Протягом багатьох років було запропоновано багато алгоритмів пошуку найближчих сусідів; вони, як правило, прагнуть зменшити кількість фактично виконаних оцінок відстані. k-NN має деякі сильні результати послідовності. Різні покращення швидкості k-NN можливі за допомогою діаграм близькості.

У k-NN регресії алгоритм k-NN використовується для оцінки безперервних змінних. Один із таких алгоритмів використовує зважене середнє значення k найближчих сусідів, зважене за величиною, оберненою їх відстані. Цей алгоритм працює наступним чином:

- Обчисліть евклідову відстань або відстань Махаланобіса від прикладу запиту до позначених прикладів.

- Упорядкуйте позначені приклади за збільшенням відстані.

- Знайдіть евристично оптимальну кількість k найближчих сусідів на основі RMSE. Це робиться за допомогою перехресної перевірки. -

Обчисліть зворотне зважене середнє значення відстані з k-найближчими багатовимірними сусідами.

Відстань до k-го найближчого сусіда також можна розглядати як локальну оцінку щільності, і, отже, також є популярним викидом у виявленні аномалій. Чим більша відстань до k-NN, тим менша локальна щільність, тим більша ймовірність, що точка запиту є викидом. Незважаючи на те, що ця модель викидів досить проста, разом з іншим класичним методом інтелектуального аналізу даних, фактором локальних викидів, працює досить добре в порівнянні з новішими та більш складними підходами, згідно з широкомасштабним експериментальним аналізом.

1.8 Content-based image retrieval technique

Метод content-based image retrieval (CBIR) - це техніка пошуку зображень, яка використовує вміст (зміст) зображення, а не метадані або ключові слова, для виконання пошуку. Один з підходів до реалізації CBIR полягає у використанні згорткових нейронних мереж (CNN) для витягування вектора характеристик зображення, а потім використовує модель перенесення для перетворення цього вектора в вектор ембедінгів, який використовується для пошуку.

Аналіз методу content-based image retrieval з використанням CNN може бути поділений на наступні кроки:

а) Передпроцесинг: Зображення піддаються передпроцесингу, щоб забезпечити стандартизацію та нормалізацію даних. Це може включати зміну розміру зображення до стандартного розміру, конвертацію в чорно-біле зображення або нормалізацію значень пікселі;

б) Витягнення ознак зображень за допомогою CNN: Застосовується попередньо навчена CNN, наприклад, VGG або ResNet, для витягування ознак зображення. Конволюційні та пулінгові шари мережі допомагають виявити локальні особливості, такі як границі, текстури та форми, а повнозв'язні шари створюють біль високорівневі характеристики зображення;

в) Векторизація ознак: Вихід з останнього повнозв'язного шару CNN є вектором ознак зображення. Зазвичай цей вектор має велику кількість компонентів. Щоб скоротити його розмірність та зберегти інформацію, використовують Max Pooling;

г) Модель перенесення знань (embedder): Вектори ознак зображень передаються через модель перенесення знань (transfer learning model) для перетворення їх в вектори ембедінгів нижчої розмірності. Ця модель може бути навчена на великому наборі даних з інших задач, наприклад, класифікації

зображень. Такий підхід дозволяє "перенести" знання, отримане моделлю на великому наборі даних, до завдання CBIR.

д) Пошук за вектором ембедінгу: При пошуку зображень за допомогою CBIR використовується відстань між векторами ембедінгів для порівняння схожості зображень. Найпростіший метод - це порівняння евклідової відстані між векторами ембедінгів. Більш складні методи можуть використовувати метрики схожості, такі як косинусна схожість.

CBIR з використанням CNN та моделі перенесення знань дозволяє витягати високорівневі характеристики зображень та створювати компактні вектори ембедінгів, що полегшує пошук та порівняння зображень. Цей підхід є потужним інструментом для широкого спектру застосувань, таких як пошук зображень за схожістю, класифікація та опис зображень.

РОЗДІЛ 2 ОПИС ТЕХНОЛОГІЙ ТА МОВ ПРОГРАМУВАННЯ

2.1 Мова Python

Python — мова програмування високого рівня загального призначення. Основна філософія дизайну Python полягає у забезпеченні читабельності коду шляхом використання значного відступу згідно правила відступів. Python є динамічно типізованою мовою з автоматичним збором сміття. Вона підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об'єктно-орієнтоване та функціональне програмування. Велику популярність Python отримав завдяки його повній стандартній бібліотеці, що призводить до його опису як мови "батарейки включені".

Розробка Python була почата Гвідо ван Россумом наприкінці 1980-х як наступник мови програмування ABC. Перша версія Python, Python 0.9.0, була випущена в 1991 році. У 2000 році було випущено Python 2.0. Основною версією стала Python 3.0, яка була випущена в 2008 році і не була повністю сумісною з попередніми версіями. Останнім релізом Python 2 був Python 2.7.18, випущений у 2020 році. Python є однією з найпопулярніших мов програмування.

Python є багатопарадигмовою мовою програмування. Вона повністю підтримує об'єктно-орієнтоване та структуроване програмування, а також має функціональні та аспектно-орієнтовані можливості (включаючи метапрограмування та метаоб'єкти). Крім того, за допомогою розширень підтримуються інші парадигми, такі як проектування за контрактом і логічне програмування. Python використовує динамічну типізацію та комбінацію підрахунку посилань і збирача сміття для керування пам'яттю. Він також використовує динамічне розпізнавання імен (пізнє зв'язування), що зв'язує імена методів і змінних під час виконання програми.

Його дизайн пропонує певну підтримку функціонального програмування в традиції Lisp. Він має функції фільтра, відображення та зменшення; списки розуміння, словники, набори та генераторні вирази.

Стандартна бібліотека має два модулі (itertools і functools), які реалізують функціональні інструменти, запозичені з Haskell і Standard ML. Його основна філософія коротко викладена в документі The Zen of Python (PEP 20), який містить такі афоризми, як:

- Красиве краще, ніж потворне.
- Явне краще, ніж неявне.
- Просте краще, ніж складне.
- Читабельність має значення.

Велика стандартна бібліотека Python надає інструменти, які підходять для вирішення багатьох завдань, і її зазвичай називають однією з її найбільших переваг.

Для додатків, що працюють в Інтернеті, підтримується багато стандартних форматів і протоколів, таких HTTP. Він включає модулі для створення графічних інтерфейсів користувача, підключення до реляційних баз даних, генерування псевдовипадкових чисел, арифметики з десятковими дробами довільної точності, маніпулювання регулярними виразами та модульного тестування.

Деякі частини стандартної бібліотеки охоплюються специфікаціями — наприклад, реалізація інтерфейсу шлюзу веб-сервера (WSGI) wsgiref відповідає PEP 333— але більшість визначено їх кодом, внутрішньою документацією та наборами тестів. Однак, оскільки більша частина стандартної бібліотеки є крос-платформним кодом Python, лише деякі модулі потребують змін або переписування для варіантних реалізацій.

Python може використовуватися як мова сценаріїв для веб-додатків, наприклад, через модуль mod_wsgi для веб-сервера Apache. За допомогою стандартного API, що працює через інтерфейс шлюзу веб-сервера, розроблено зручний фреймворк для полегшення роботи з цими програмами. Для розробки та обслуговування складних програм розробники використовують такі веб-фреймворки, як Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle і Zope. Для розробки клієнтських програм на базі Ajax можна використовувати

Pyjs і IronPython. SQLAlchemy дозволяє використовувати Python як інструмент відображення даних у реляційну базу даних. Twisted є платформою для програмування зв'язку між комп'ютерами, і вона використовується, наприклад, в Dropbox.

У наукових обчисленнях Python ефективно використовується з такими бібліотеками, як NumPy, SciPy і Matplotlib. Для конкретних областей існують спеціалізовані бібліотеки, такі як Biopython і Astropy, які допомагають виконувати наукові обчислення відповідно до потреб користувача. SageMath є системою комп'ютерної алгебри з інтерфейсом ноутбука, розробленим на Python, і має багато математичних можливостей, таких як алгебра, комбінаторика, чисельна математика, теорія чисел і числення.

OpenCV має прив'язки Python і надає широкий набір функцій для комп'ютерного зору та обробки зображень.

Python широко використовується в проектах штучного інтелекту та машинного навчання з використанням бібліотек, таких як TensorFlow, Keras, PyTorch і scikit-learn. Завдяки своїй модулярній архітектурі, простому синтаксису та інструментам для обробки форматованого тексту, Python часто використовується для обробки природної мови. Також Python можна використовувати для створення ігор за допомогою бібліотек, наприклад, Pygame, які дозволяють створювати 2D-ігри.

2.2 Бібліотека PyTorch

PyTorch є платформою машинного навчання, яка базується на бібліотеці Torch. Вона використовується для розв'язання завдань, таких як комп'ютерне бачення та обробка природної мови. Спочатку PyTorch була розроблена компанією Meta AI, а зараз вона є частиною парасольки Linux Foundation. Це безкоштовне

програмне забезпечення з відкритим кодом, яке поширюється за модифікованою ліцензією BSD.

PyTorch має розширений інтерфейс Python, який є основним напрямком розробки, але також підтримує інтерфейс C++. З використанням PyTorch було створено багато програм для глибокого навчання, зокрема Tesla Autopilot, Pyro від Uber, Transformers від Hugging Face, PyTorch Lightning і Catalyst.

PyTorch надає дві важливі високорівневі функції: тензорні обчислення (аналогічні до NumPy) з потужним прискоренням за допомогою графічних процесорів (GPU) та побудову глибоких нейронних мереж на основі системи автоматичного диференціювання.

Основним пакетом в PyTorch є torch, який надає гнучкий N-вимірний масив або тензор. Цей об'єкт підтримує базові операції, такі як індексування, нарізання, транспонування, зміна розміру, приведення типів, спільне використання пам'яті та клонування. Він є основним елементом бібліотеки і використовується багатьма іншими пакетами. Тензор також підтримує математичні операції, такі як максимум, мінімум, сума, а також різні статистичні розподіли, наприклад, рівномірний, нормальний і мультиноміальний. Крім того, він підтримує операції BLAS, такі як скалярний добуток, множення матриці на вектор, множення матриць та матричний добуток.

2.3 Torchvision

Torchvision - це популярний пакет у PyTorch, який надає доступ до різноманітних наборів даних, моделей та перетворень зображень для завдань комп'ютерного зору. Він призначений для спрощення створення моделей глибокого навчання для аналізу зображень та відео.

Пакет torchvision містить широкий набір попередньо навчених моделей, таких як AlexNet, VGG, ResNet та інші, які можна легко використовувати для завдань, таких як класифікація зображень, виявлення об'єктів та семантична сегментація. Ці моделі були навчені на великомасштабних наборах даних, таких

як ImageNet, і можуть бути доналаштовані або використовуватись для передачі навчання. Крім того, torchvision містить набір наборів даних, які часто використовуються у дослідженнях з комп'ютерного зору, таких як MNIST, CIFAR-10 та COCO. Ці набори даних можна завантажити безпосередньо в код PyTorch, що дозволяє швидко почати експериментувати з моделями глибокого навчання.

Крім цього, torchvision надає різноманітні перетворення зображень, включаючи зміну розміру, обрізку, нормалізацію та методи аугментації даних. Ці перетворення є необхідними для попередньої обробки зображень перед їх подачею на вхід моделям глибокого навчання.

Загалом, torchvision - це цінний інструмент для роботи з завданнями комп'ютерного зору у PyTorch, який надає готові до використання моделі, набори даних та перетворення зображень, що значно спрощують процес розробки.

2.4 Faiss

Faiss(Facebook AI Similarity Search) - одна з найпопулярніших реалізацій ефективного пошуку подібності. Faiss вимірює L2 (або евклідову) відстань між усіма заданими точками між нашим вектором запиту та векторами, завантаженими в індекс, для наступного швидкого пошуку.

2.5 Середовище розробки Microsoft Visual Studio

Microsoft Visual Studio є набором продуктів, розроблених компанією Microsoft, які включають інтегроване середовище розробки програмного забезпечення та інші інструменти. Ці продукти дозволяють розробляти консольні додатки, ігри та програми з графічним інтерфейсом, включаючи підтримку технології Windows Forms, а також веб-сайти, веб-додатки та веб-служби у різних платформах, таких як Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone, .NET Compact Framework і Silverlight. Microsoft Visual

Studio 2019 використовується як основне середовище розробки. Воно включає редактор вихідного коду з підтримкою технології IntelliSense і можливість простого рефакторингу коду.

Також містить вбудований відладчик, який працює на рівні вихідного коду та машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio також дозволяє підключати сторонні додатки (плагіни), що розширюють його функціональність на різних рівнях, включаючи підтримку систем контролю версій вихідного коду (такі як Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування та візуального проектування коду на мовах програмування з орієнтацією на об'єкти) або інструментів для інших аспектів процесу розробки програмного забезпечення.

2.6 OpenCV

OpenCV - це бібліотека функцій програмування, головним чином для комп'ютерного зору в реальному часі. Спочатку розроблений Intel, пізніше він був підтриманий Willow Garage, потім Itseez (який пізніше був придбаний Intel). Бібліотека є кросплатформною та ліцензована як безкоштовне програмне забезпечення з відкритим вихідним кодом згідно з ліцензією Apache License 2. Починаючи з 2011 року OpenCV підтримує прискорення GPU для операцій у реальному часі.

OpenCV написаний на мові програмування C++, як і його основний інтерфейс, але він все ще зберігає менш повний, але розширений старий інтерфейс C. Усі нові розробки та алгоритми з’являються в інтерфейсі C++. Існують прив’язки до мов у Python, Java та MATLAB/Octave. Інтерфейс прикладного програмування (API) для цих інтерфейсів можна знайти в онлайн-документації. Бібліотеки-оболонки кількома мовами були розроблені, щоб заохотити прийняття ширшою аудиторією. У версії 3.4 прив’язки JavaScript для

вибраної підмножини функцій OpenCV було випущено як OpenCV.js для використання на веб-платформах.

Якщо бібліотека знаходить інтегровані примітиви продуктивності Intel у системі, вона використовуватиме ці власні оптимізовані процедури для прискорення. Інтерфейс графічного процесора (GPU) на основі Compute Unified Device Architecture (CUDA) розробляється з вересня 2010 року. Інтерфейс GPU на основі OpenCL розробляється з жовтня 2012 року.

2.7 Numpy

Рисунок 2.1 Логотип NumPy

NumPy - це бібліотека для мови програмування Python, яка надає можливості роботи з великими багатовимірними масивами і матрицями, а також велику колекцію математичних функцій високого рівня для операцій з цими масивами. (Рисунок 2.1) Початково, Numeric була попередньою версією NumPy, розробленою Джимом Хугуніним та іншими розробниками. У 2005 році Тревіс Оліфант створив NumPy, включивши функціональність конкуруючої бібліотеки Numarray зі значними змінами. NumPy є відкритим програмним забезпеченням з великою кількістю учасників, і його розвиток фінансується організацією NumFOCUS.

NumPy спрямований на оптимальну реалізацію для стандартного інтерпретатора Python - CPython, який є неоптимізованим інтерпретатором байт-коду. Алгоритми, написані для Python, часто працюють повільніше, ніж їх скомпільовані еквіваленти через відсутність оптимізації компілятора. NumPy частково вирішує цю проблему, надаючи ефективні масиви, функції та оператори для роботи з масивами. Використання NumPy вимагає переписування коду, зокрема внутрішніх циклів, для використання його функціональності. Використання NumPy у Python надає схожу функціональність з MATLAB, оскільки обидва інтерпретуються і дозволяють швидку обробку даних в масивах та матрицях. MATLAB має багато додаткових інструментів, таких як Simulink, тоді як NumPy входить у склад Python як більш сучасна та повнофункціональна мова програмування. Крім того, доступні додаткові пакети Python, такі як SciPy, який надає більше можливостей, подібних до MATLAB, і Matplotlib, який забезпечує функціональність для побудови графіків, схожу на MATLAB.

Як MATLAB, так і NumPy покладаються на оптимізовані бібліотеки лінійної алгебри BLAS і LAPACK для ефективних обчислень. NumPy також використовується в прив'язках Python до популярної бібліотеки комп'ютерного зору OpenCV для зберігання та роботи з зображеннями. Використання масивів NumPy спрощує обробку зображень, так як зображення можуть бути представлені як тривимірні масиви, і масиви NumPy надають ефективні методи доступу до пікселів зображення шляхом індексування, нарізки або маскування. Основна функціональність NumPy полягає у структурі даних "ndarray" для багатовимірних масивів. Ці масиви є послідовним видом у пам'яті і мають однорідний тип даних. Вони також можуть бути представлені як представлення буферів пам'яті, виділених іншими мовами програмування, такими як C/C++, Python і Fortran, без необхідності копіювання даних, що забезпечує сумісність з існуючими числовими бібліотеками.

2.8 Pillow

Pillow - бібліотека мов Python, призначена для роботи з растровою графікою. Включає, в тому числі, підтримку Python 3.x. Цей форк прийнято в якості заміни оригінальної бібліотеки та включено до деяких дистрибутивів Linux, включаючи Debian і Ubuntu.

2.9 Matplotlib

Matplotlib - це бібліотека для побудови графіків для мови програмування Python та її розширення чисельної математики NumPy. Він надає об’єктно-орієнтований API для вбудовування графіків у програми за допомогою наборів інструментів загального призначення GUI, таких як Tkinter, wxPython, Qt або GTK. Існує також процедурний інтерфейс "pylab", заснований на кінцевій машині (як OpenGL), розроблений, щоб дуже нагадувати інтерфейс MATLAB, хоча його використання не рекомендується. SciPy використовує Matplotlib. Matplotlib спочатку був написаний Джоном Д. Хантером. З тих пір вона має активну спільноту розробників і поширюється за ліцензією у стилі BSD. Майкл Дреттбум був призначений провідним розробником matplotlib незадовго до смерті Джона Хантера в серпні 2012 року, а потім до нього приєднався Томас Касвелл.. Matplotlib є фінансово фінансованим проектом NumFOCUS. Matplotlib 2.0.x підтримує Python версій від 2.7 до 3.10. Підтримка Python 3 почалася з Matplotlib 1.2. Matplotlib 1.4 є останньою версією, яка підтримує Python 2.6. Matplotlib пообіцяв не підтримувати Python 2 після 2020 року, підписавши Заяву про Python 3.

Доступно кілька наборів інструментів, які розширюють функціональність Matplotlib. Деякі є окремими завантаженнями, інші постачаються з вихідним кодом Matplotlib, але мають зовнішні залежності. Базова карта: побудова карти з різними проекціями карти, береговими лініями та політичними кордонами

Cartopy: бібліотека відображень, що містить визначення проекцій об’єктно-орієнтованої карти та можливості трансформації довільних точок, ліній,

багатокутників і зображень. (Matplotlib v1.2 і вище)

Інструменти GTK: інтерфейс до бібліотеки GTK Інтерфейс Qt

Mplot3d: тривимірні графіки

Natgrid: інтерфейс до бібліотеки natgrid для сітки нерівномірно розташованих даних.

tikzplotlib: експорт до Pgfplots для плавної інтеграції в документи LaTeX (раніше відомий як matplotlib2tikz)

Seaborn: надає API на основі Matplotlib, який пропонує розумний вибір стилю графіка та кольорів за замовчуванням, визначає прості високорівневі функції для поширених типів статистичних графіків та інтегрується з функціями, наданими Pandas

2.10 Tkinter

Бібліотека Tkinter (що розшифровується як "Tk інтерфейс") є стандартною бібліотекою для створення графічного інтерфейсу користувача (GUI) у мові програмування Python. Tkinter базується на Tk, яка є набором інструментів для розробки інтерфейсу користувача, створеною для мови Tcl (Tool Command Language). Tkinter дозволяє створювати вікна, кнопки, поля введення, мітки та інші елементи GUI, які взаємодіють з користувачем.

Однією з головних переваг Tkinter є те, що вона входить у стандартну бібліотеку Python, тому не потребує окремого встановлення. Вона доступна на різних платформах, включаючи Windows, macOS і Linux, що робить її популярним вибором для розробки користувацьких інтерфейсів в Python.

Tkinter надає набір класів та методів для створення та керування елементами GUI. Ви можете створювати вікна і розміщувати на них кнопки, мітки, поля введення, списки та інші елементи. Ви можете відповідати на події, такі як натискання кнопок або рух миші, і виконувати певні дії відповідно до цих подій. Tkinter також підтримує розміщення елементів за допомогою різних

менеджерів компонування, таких як grid, pack і place, що дозволяє гнучко керувати розташуванням елементів у вікні.

Загалом, Tkinter є потужним інструментом для розробки GUI додатків в мові програмування Python. Вона надає зручний і простий спосіб створити інтерактивний інтерфейс для ваших програм, що дозволяє користувачеві взаємодіяти з вашим кодом у візуальному середовищі.

РОЗДІЛ 3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Огляд датасету

Після аналізу низки датасетів, доступних в Інтернеті, було вирішено використати Scool lunch dataset[16] як основний джерело даних для дослідження. Цей датасет вражає своєю обширністю та розмаїттям інформації, що в ньому міститься. Кожен файл у цьому датасеті містить фотографію шкільного обіду та детальний опис, що супроводжує зображення.

Кожен опис включає повний перелік об'єктів, що присутні на фотографії, а також надається класифікація цих об'єктів разом із їхніми координатами на зображенні. Така інформація надає унікальну можливість дослідити різноманітні аспекти шкільних обідів, відслідковуючи конкретні продукти та їх розташування на тарілках. Scool lunch dataset містить в собі 3940 фотографій, що забезпечує широкий обсяг візуальних даних. Крім того, цей датасет включає 21 унікальний клас, що визначає різні категорії продуктів та страв, що присутні на зображеннях (Рисунок 3.1).

Список усіх класів: Milk, Drinkable yogurt, Rice, Mixed rice, Bread,White bread, Udon, Fish, Meat, Salad, Cherry tomatoes, Soups, Curry, Spicy chili-flavored tofu, Bibimbap, Fried noodles, Spaghetti, Citrus, Apple, Cup desserts, Other foods.

Рисунок 3.1 Приклад одного анотованного фото з датасету

3.2 Обробка датасету

Для роботи з кожним класом треба отримати окремі фото. Був написаний код для обрізання блюд з фото і збереження в папки по класах (Рисунок 3.2)

```
def read_annotations(file_path):
    lines = open(file_path).readlines()
    data = []
    for l in lines:
        cls, x1, y1, x2, y2 = [int(i) for i in l.split()]
        data.append(
            {
                'box': [x1, y1, x2, y2],
                'class': cls
            }
        )
    return data

def path_to_id(path):
    return path.split('/')[-1].split('.')[0]

id_to_annotation = {
    path_to_id(p): read_annotations(p)
    for p in glob("school_lunch/Annotations/*")}

for image_path in tqdm(glob("school_lunch/Images/*")):
    id_ = path_to_id(image_path)
    img = cv2.imread(image_path)

    for i, bbox in enumerate(id_to_annotation[id_]):
        x1, y1, x2, y2 = [max(0, i) for i in bbox['box']]
        object_image = img[y1:y2, x1:x2]
        cls = bbox['class']
        save_path = f'school_lunch/cropped/{cls}/'
        object_path = save_path + f'{id_}_{i}.jpg'
        if os.path.exists(object_path):
            break
        os.makedirs(save_path, exist_ok=True)
        cv2.imwrite(save_path + f'{id_}_{i}.jpg', object_image)
```

Рисунок 3.2 Код для обрізання фотографій по координатам

`def read_annotations(file_path) –`
Поверне масив списків з інформацією про блюда на фотографії

3.3 Отримання вектору характеристик з зображень за допомогою VGG16

Завантажуємо бібліотеку torchvision і достаємо модель(Рисунок 3.3)

```
import torchvision as tv
vgg16 = tv.models.vgg16(weights=tv.models.VGG16_Weights.DEFAULT)
```

Рисунок 3.3 Завантаження VGG16 моделі у пам’ять

Отримаємо вектору характеристик з тренувального сету (Рисунок 3.4)

```
@torch.no_grad()
def get_features(model, X):
    X = model.features(X)
    X = F.avg_pool2d(X, kernel_size=X.shape[-2:])[..., 0, 0]
    return X
Run Cell | Run Above | Debug Cell
#%%
epoch_to_features = {}

vgg16.eval()
for e in range(50):
    features = []
    labels = []
    for images, cls in tqdm(train_loader):
        batch_features = get_features(vgg16, images.to(device)).cpu().numpy()
        features.append(batch_features)
        labels.extend(cls)
    features = np.concatenate(features)

    epoch_to_features[e] = {
        'features': features,
        'labels': np.array(labels, int)
    }
```

Рисунок 3.4 Код для отримання вектору характеристик з зображення

Ініціалізуємо TripletLoss з функцією дистанції (Рисунок 3.5)

```
class TripletLoss(nn.Module):
    def __init__(self, margin=1.0):
        super(TripletLoss, self).__init__()
        self.margin = margin

    def calc_euclidean(self, x1, x2):
        return torch.sqrt((x1 - x2).pow(2).sum(1))

    def forward(self, anchor, positive, negative):
        distance_positive = self.calc_euclidean(anchor, positive)
        distance_negative = self.calc_euclidean(anchor, negative)
        losses = torch.clamp(distance_positive - distance_negative + self.margin, min=0)
        return losses.mean()
```

Рисунок 3.5 Код з описом класу TripletLoss

3.4 Етап розпізнавання

Для пошуку схожого зображення використовується бібліотека faiss. Використовується клас IndexFlatL2 (Індексер) з faiss для зберігання та пошуку ембедінгів. Завантажуємо всі ембедінги з тренувального сету в індексер.

Для кожного зображення, яке потрібно розпізнати, ми витягуємо вектор характеристик з VGG16. Потім, використовуючи натреновану модель (embedder), ми отримуємо його ембедінг. За допомогою ембедінгу зображення ми шукаємо найближчий ембедінг в індексері. Клас зображення, якому належить найближчий ембедінг, буде використаний для передбачення.(Рисунок 3.6).

```
def predict_class(model, embedding_size, feature_loader, photo_paths: list):
    model.eval()
    index, train_lables = load_indexer(model, embedding_size, feature_loader)
    result = []
    for path in photo_paths:
        emb = model.forward_from_features(get_features(vgg16, path)[None])
        _, I = index.search(emb.cpu().numpy(), 1)
        indecies = np.take(train_lables, I)
        result.append(indecies)
    return result
```

Рисунок 3.6 Етап розпізнавання

3.5 Валідація моделі

Для кожного зображення в валідаційному сеті додаємо текст з класом що передбачила модель (Рисунок 3.7).

```
for i, (photo, predicted_class) in enumerate(zip(photos_to_predict, predictions)):
    img = cv2.imread(photo)
    img = cv2.resize(img, (300, 300))
    font                   = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (10,30)
    fontScale              = 0.8
    fontColor              = (0,0,0)
    thickness              = 2
    lineType               = 2

    cv2.putText(img,
                categories[predicted_class],
        bottomLeftCornerOfText,
        font,
        fontScale,
        fontColor,
        thickness,
        lineType)

    plt.imshow(img[..., ::-1])
    plt.show()
    target = int(photo.split('cropped/')[-1].split('/')[0])
    print(f"Predicted label {categories[predicted_class]}")      You, 4 days ago • init …
    print(f"RealLabel label {categories[target]}")
```

Рисунок 3.7 Код для нанесення тексту з на зображення

Після тренерування можемо протестувати на валідаційному сеті, отримаємо результати (Рисунок 3.8)

Рисунок 3.8 Тестові зображення з передбаченням моделі

Після аналізу передбачень моделі на всіх тестових зображеннях, прототип показує гарні результати досягаючи 98.3% точності.

3.6 Опис роботи програми

Після запуску програми, відкривається вікно. Для вибору фотографії треба натиснути кнопку “Upload File” і вибрати фотографію, після цього фотографія вібразиться на вікні.

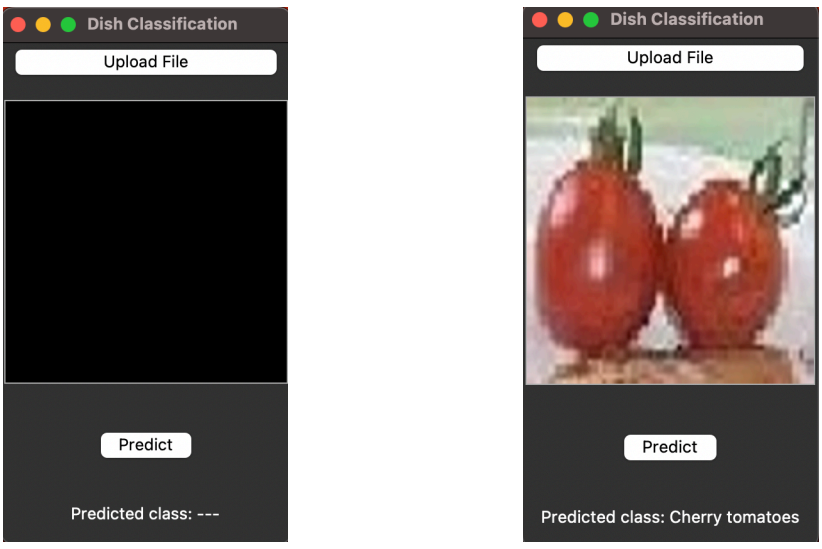

Рисунок 3.8 Інтерфейс програми

Для класифікації треба натиснути кнопку “Predict”, після чого з’явиться текст з класом блюда. (Рисунок 3.8)

ВИСНОВКИ

Провідні методи штучного інтелекту в комп'ютерному можуть дуже покращити швидкість обслуговування за допомогою класифікації блюд на тацях у закладах харчування за допомогою фотографій.

Використання ембеденгів та алгоритмів глибокого навчання дозволяє автоматично розпізнавати та класифікувати різні види їжі, що прискорює процес обробки та замовлення їжі.

Застосування цієї технології у закладах харчування, де оплата здійснюється на касі, сприяє зручності та ефективності обслуговування. Клієнти можуть просто розмістити свої тарілки на таці, а система автоматично ідентифікує блюда та обчислює вартість замовлення. Це дозволяє уникнути черг та зайвої взаємодії з персоналом, що полегшує процес придбання їжі та зменшує час очікування.

Класифікація блюд за допомогою ембеденгів також забезпечує високу точність і надійність ідентифікації різних страв. Алгоритми глибокого навчання можуть враховувати різноманітні особливості, такі як форма і колір, що дозволяє забезпечити точну класифікацію навіть у випадку схожих блюд. Це сприяє уникненню помилок та покращує задоволення клієнтів. Загалом, класифікація блюд за допомогою ембеденгів на тацях у закладах харчування з оплатою на касі є інноваційним підходом, який сприяє поліпшенню ефективності та якості обслуговування. Ця технологія не тільки прискорює процес замовлення та оплати, але й забезпечує більш точну та зручну ідентифікацію різних страв. Впровадження цього рішення може принести значні переваги як для закладів харчування, так і для клієнтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A Comprehensive Guide to Convolutional Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

2. Трансферне навчання в нейроних мережах - [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/

3. VGG16 - [Електронний ресурс] – Режим доступу до ресурсу: https://builtin.com/machine-learning/vgg16

4. Опис VGG16 - [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

5. Імплементація VGG16. - [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/api/applications/vgg/

6. Triplet Loss - [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/triplet-loss-advanced-intro-49a07b7d8905

7. Алгоритм k-найближчих сусідів - [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e

8. PyTorch - [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/

9. Сторінка Python - [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/

10. Faiss - [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/getting-started-with-faiss-93e19e887a0c

11. Faiss код - [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/facebookresearch/faiss

12. Opencv код - [Електронний ресурс] – Режим доступу до ресурсу: https://opencv.org/

13. Код Numpy - [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/

14. VGG16 - [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1409.1556.pdf

15. Deep metric learning using triplet network - [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1412.6622.pdf

16. Food lunch dataset - [Електронний ресурс] – Режим доступу до ресурсу: http://foodcam.mobi/dataset.html