МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра прикладної математики

Дипломна робота

Дослідження розпізнавання об'єктів з використанням TensorFlow

Виконав: студент групи ПМП-42
спеціальності
113 - прикладна математика

_____Мамчур О.О._____
(прізвище та ініціали)

Керівник _____Дяконюк Л.М._____
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Львів - 2023

ЗМІСТ

ВСТУП

Метою дипломної роботи є дослідження виявлення та розпізнавання облич на зображенні. У цій роботі **предметом дослідження** є методи для роботи з зображеннями облич, використовуючи згорткові нейронні мережі (Convolutional Neural Networks, CNN).

За останні кілька років можна спостерігати значний розвиток області обробки зображень, зокрема, використання згорткових мереж для розпізнавання облич. Розпізнавання облич має великий потенціал у багатьох сферах, включаючи безпеку, автоматичну ідентифікацію, відеоспостереження, розваги та соціальні медіа.

Слід знати про проблеми, з якими можна зіштовхнутися під час дослідження цієї сфери.

- Різноманітність облич. Людські обличчя можуть сильно відрізнятися за формою, розташуванням органів (наприклад, очі, ніс, рот). Це призводить до труднощів виявлення.
- Елементи, що схожі на обличчя. Деякі об’єкти можуть виглядати як справжні обличчя. Це може призвести до помилкових результатів, коли система виявляє об’єкти, які не є справжніми обличчями.
- Освітлення. Обличчя можуть бути затемненими, а рівень освітлення може змінюватися. Це ускладнює процес визначення обличчя та знижує точність системи.
- Розмір і масштаб. Обличчя можуть відрізнятися за розміром і масштабом на зображеннях, що ускладнює їх виявлення.

- Розташування та орієнтація. Обличчя можна розташовувати під різними кутами та орієнтаціями на зображенні. Для цього можуть знадобитися додаткові методи виявлення та корекції орієнтації облич.
- Обробка в реальному часі. Вимога до обробки в реальному часі може бути проблемою, оскільки система повинна працювати швидко й ефективно.

У цій роботі буде розглянуто різні зображення людини, де буде змінюватися, ракурс, кут, відстань, освітлення. Буде досліджено використання CNN для роботи з цими зображеннями.

ФОРМУЛЮВАННЯ ЗАДАЧІ

Вважатимемо, що вхідними даними для досліджень буде виступати зображення (фото), на якому можуть бути зображені різні об'єкти, в тому числі й люди. У найбільш загальному випадку фото може бути отримано, як окремий кадр з відеопотоку. Фото може бути задано в різних форматах і вважатимемо, що воно пройшло попередню обробку за різними аспектами: очищення від шуму, контрастування і так далі. Також, попередньо зображення слід нормалізувати, а саме, привести до формату, який здатна опрацьовувати запропонована алгоритмом розвязування модель. Задача полягає в знаходженні та виділенні облич людей, зазвичай це роблять за допомогою прямокутників конкретного кольору. В подальшому знайдені обличчя людей слід ідентифікувати на приналежність до конкретної бази даних.

1. ПОРІВНЯННЯ АЛГОРИТМІВ

Виявлення та розпізнавання обличчя на зображенні, використовуючи машинне навчання є важливою задачею в сфері комп'ютерного зору і обробки зображень. Існує багато алгоритмів для вирішення цих проблем. Наведемо декілька найпопулярніших з них і опишемо їхні властивості [1].

1.1 Eigenfaces з алгоритмом PCA

У 1991 році Терк і Пентланд запропонували підхід [2] до розпізнавання облич з використанням концепцій зменшення розмірності та лінійної алгебри. PCA (Principal Component Analysis) - це метод зменшення розмірності, запропонований Пірсоном у 1901 році. Він використовує власні значення та власні вектори для зменшення розмірності та проектування навчальної вибірки/даних на менший простір ознак.

Вважатимемо, що є набір з m зображень розмірності $n * n$

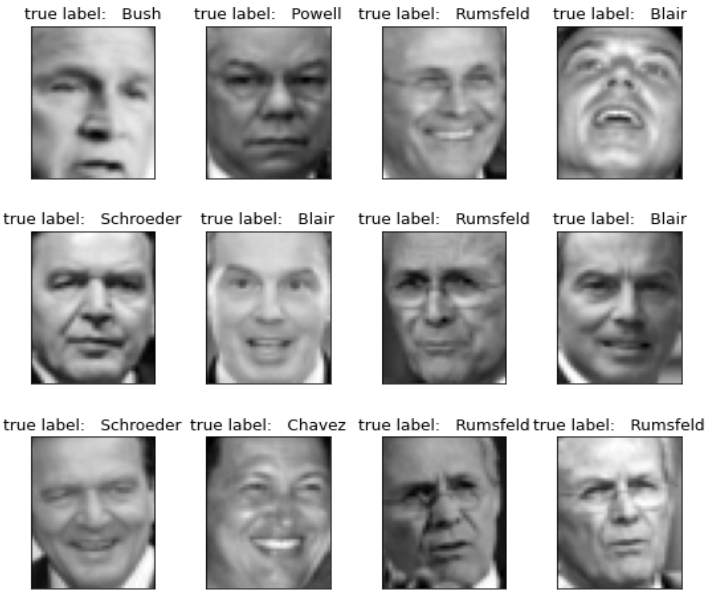


Рис 1. Тестові зображення з мітками

Змінимо розмірність вектора $n * n$, що відповідає зображенню, у вектор розмірності $n^2 * 1$

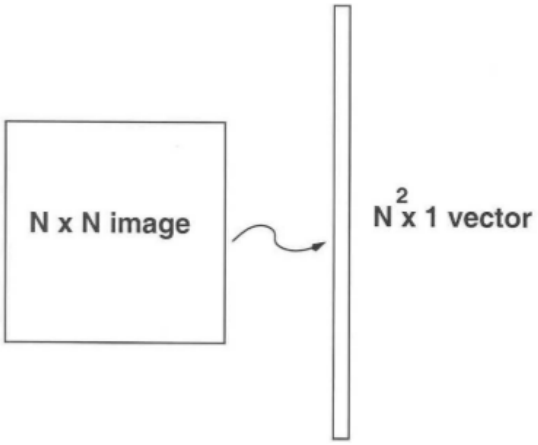


Рис. 2 Перетворення зображення у вектор

Тепер обчислимо середнє значення всіх цих векторів облич і будемо віднімати його від кожного вектора $x_1, x_2, x_3, \ldots x_m$

$$\psi = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$a_i = x_i - \psi$$

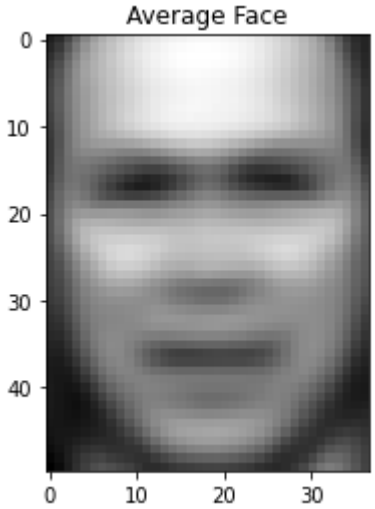


Рис 3. Зображення після віднімання середнього значення

Отримаємо матрицю A розмірності $n^2 * m$

$$A = [a_1\ a_2\ a_3 \ldots\ a_m]$$

Знайдемо коваріантну матрицю $C_{ov} = A^T * A$

Після цього обчислимо власні значення та власні вектори за формулою

$$A^T A v_i = \lambda_i v_i$$

$$A A^T A v_i = \lambda_i A v_i$$

$C' u_i = \lambda_i u_i$, де $C' = A A^T$ і $u_i = A v_i$

Тепер обчислимо власний вектор і власні значення коваріантної матриці та будемо відображати їх у C' за допомогою формули $u_i = A v_i$

На цьому кроці використовуємо власні вектори отримані на попередньому кроці. Будемо брати нормалізовані навчальні обличчя (face - середнє обличчя) x_i і представляємо вектори кожного обличчя у вигляді лінійної комбінації найкращих K власних векторів.

$$a_i = \sum_{j=1}^{K} \omega_j u_j$$

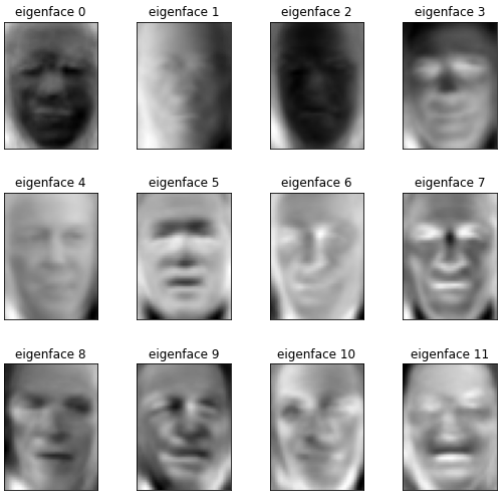


Рис 4. Власні значення (EigenFaces)

Беремо коефіцієнт власних значення і представляємо навчальні зображення облич у вигляді вектора цих коефіцієнтів.

$$x_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ . \\ . \\ w_k^i \end{bmatrix}$$

Рис. 5 Лінійна комбінація

Після чого, можна перетворити кожне вхідне зображення за допомогою власних значень. І порівнювати зображення за певною нормою, яка покаже чи обличчя схожі.

1.2 Viola-Jones

Viola-Jones [4] - це підхід до швидкого виявлення об’єктів. Це перший алгоритм, якому вдалося виявити об’єкти в реальному часі. Алгоритм Viola-Jones спершу визначає обличчя у градаціях сірого, а потім знаходить розташування у кольоровому зображенні.

Viola-Jones алгоритм окреслює рамку певного розміру і шукає в ній обличчя (пошук Гаарівських ознак). Після чого переміщає цю рамку на декілька пікселів в залежності від заданих параметрів.

1.2.1 Ознаки подібні на Гаарівські

Ознаки подібні на Гаарівські були названі на честь Альфреда Гаара, угорського математика в 19 столітті, який розробив концепцію вейвлетів Гаара (предок подібних до ознак Гаара).

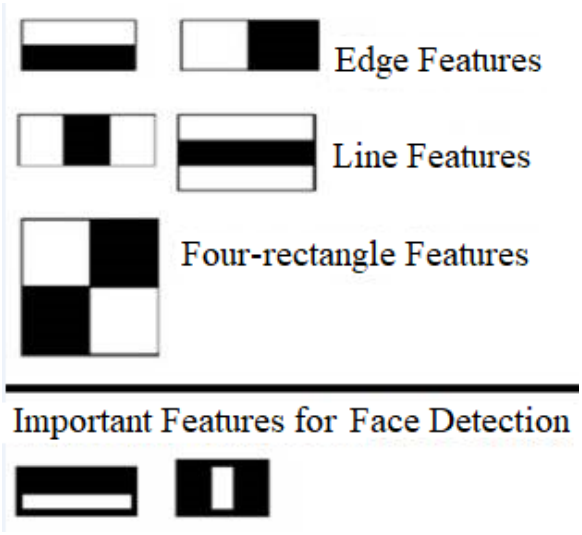


Рис. 6 Приклади ознак Гаара

У своєму дослідженні Віола та Джонс виявили 3 типи ознак, подібних до Гаарівських:

- Ознаки краю
- Ознаки лінії
- Чотиристоронні ознаки

Для виявлення обличчя важливі горизонтальна та вертикальна ознаки для розпізнавання носу, брів.

Рис. 7 Пошук брів та носу за ознаками Гаара

1.2.2 Інтегральне зображення

Для зниження затрат на обчислення ознак подібних до Гаарівських, потрібно обчислити інтегральне зображення. Щоб обчислити значення для певного пікселя, слід додати всі значення пікселів, які розташовані ліворуч нього.

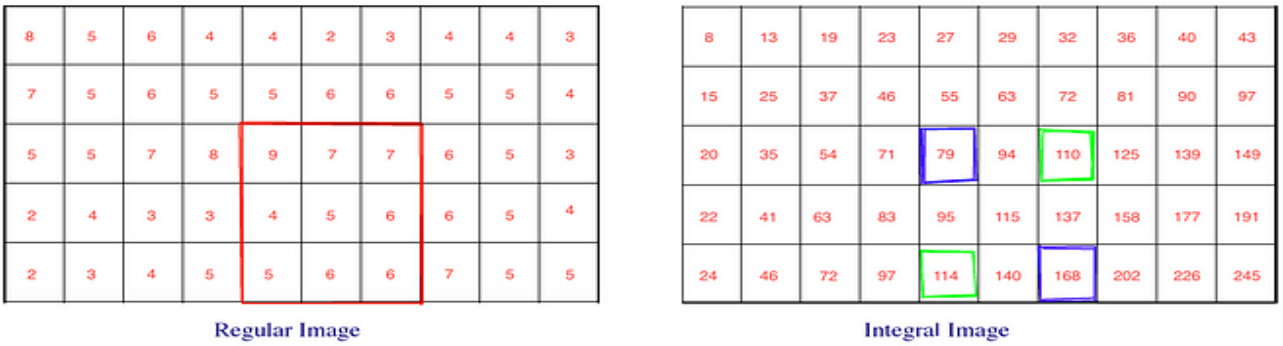


Рис. 8 Приклад побудови інтегрального зображення

Інтегральні зображення є корисними, оскільки ознаки подібні до Гаарівських є прямокутниками, і потрібно відняти просто два прямокутниками на звичайному зображені, тому що квадрати вже просумовані.

1.2.3 Алгоритм AdaBoost

Щоб навчити алгоритм розпізнавати обличчя, потрібно мати зображення, які будуть позначені так, що на них є обличчя, і зображення на яких немає облич або обличчя не справжні, і позначити так, що на них немає облич.

Алгоритм вивчає зображення і може визначати помилкові спрацьовування та несправжні дані, що дозволяє бути більш точними. Можна отримати дуже точну модель, коли буде переглянуто всі можливі позиції та комбінації цих функцій. Навчання може бути надзвичайно об'ємним через різноманітні можливості та комбінації, які доведеться перевіряти для кожного кадру чи зображення.

$$F(x) = a_1 f_1(x) + a_2 f_2(x) + a_3 f_3(x) + \dots ,$$

де $F(x)$ - це функція, яка визначає успішність (сильний класифікатор)

f_1, f_2, f_3 - Гаарівські ознаки (слабкий класифікатор)

a_1, a_2, a_3 - ваги функцій

Щоб дізнатися, які ознаки є найважливішими, потрібно ввести адаптивне посилання (Adaptive Boosting). Якщо модель не дає нам коректних результатів, адаптивне посилення доповнює поточну найкращу ознаку. Тому шукається не друга найкраща ознака, а така, що доповнить поточну найкращу ознаку. Він підвищує важливість зображень, в яких зробив помилкові висновки, і знаходить наступну найкращу ознаку, яка підійде до цих зображень, збільшуючи вагу цих

зображень в алгоритмі. Зрештою, по міру додавання нових ознак, ми знайдемо одне зображення, якому надаватиметься найбільша вага.

1.2.4 Каскадування

Каскадування — це ще один спосіб, щоб підвищити швидкість і точність моделі. Отже, будемо починати з того, що візьмемо певне підвікно, після чого, буде братися найкраща ознака та перевірятися, чи присутня вона на зображенні у підвікні. Якщо її немає у підвікні, то це вікно буде відкинуто. Якщо вона присутній, дивитимемося на другу функцію у підвікні. Якщо його немає, то це підвікно відкидається. Будемо проходити по кількості наявних ознак і будемо підвікна без них.

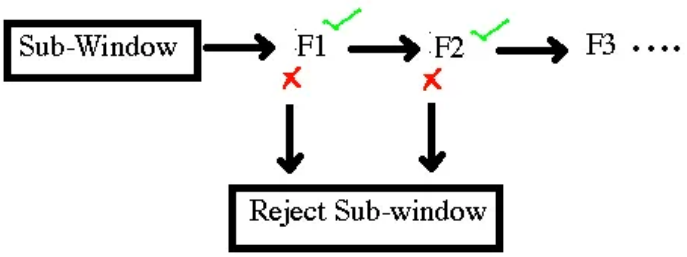


Рис. 9 Каскадування

1.3 Згорткові нейронні мережі

Згорткові нейронні мережа (Convolutional Neural Network, CNN) [6] - це нейронна мережа прямого поширення, що застосовується для аналізу зображень. Вона була розроблена спеціально для розпізнавання ознак та властивостей, які залежать від просторового розташування пікселів.

Основною особливістю CNN є використання шарів згортки, які здійснюють локальні згортки над вхідними даними з використанням фільтрів. Ці фільтри вчаться виявляти різні ознаки та шаблони в даних, такі як риси обличчя.

Основні компоненти CNN включають:

- Шари згортки (Convolutional layers). Ці шари застосовують фільтри до вхідних даних, виконуючи згортку та операцію активації, що дозволяє виявити локальні ознаки та шаблони.
- Шари пулінгу (Pooling layers). Ці шари зменшують розмірність зображення, вибираючи найбільш значущі значення з певних областей. Зазвичай використовується максимальний пулінг або середнє значення.
- Шари активації (Activation layers). Після згортки та пулінгу до результату застосовують функцію активації, яка надає нелінійність та допомагає моделі виразніше моделювати складні залежності.
- Повнозв'язані шари (Fully Connected layers). Ці шари виконують класифікацію або регресію на основі виявлених ознак. Вони з'єднуються з останнім шаром активації та генерують вихідні результати.
- Функції втрати (Loss functions). Використовуються для оцінки різниці між прогнозованими значеннями та правильними мітками та визначають, як модель навчається покращувати свої прогнози.

CNN забезпечує автоматичне виявлення ознак та ієрархічну обробку інформації, що дозволяє в них досягати вражаючих результатів у багатьох завданнях комп'ютерного зору, таких як класифікація зображень, об'єктне розпізнавання, виявлення облич та багато інших.

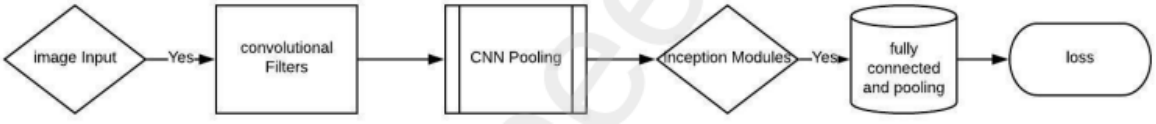


Рис 10. Структура CNN

2. МЕТОДИ РОЗВ'ЯЗАННЯ

Мій вибір для розв'язування задач виявлення та розпізнавання облич впав на алгоритм CNN. Його первагами є:

- Робота у різних умовах: CNN є більш універсальним підходом до розпізнавання облич і може бути ефективнішим за різних умов, зокрема при зміні освітлення, положення обличчя та складного фону.
- Гнучкість на адаптивність: CNN навчаються на великих обсягах даних і тому можуть розпізнавати обличчя з високою точністю. Їх також можна використовувати для інших завдань, таких як розпізнавання емоцій, рис обличчя та орієнтації, оскільки вони здатні гнучко вивчати різні ознаки.
- Локальні залежності: CNN мають вбудовану здатність розпізнавати локальні залежності на зображеннях за допомогою згортки шарів та об'єднання, ефективніше моделюючи просторові структури, такі як обличчя.

Підсумовуючи, CNN - це чудовий алгоритм, для роботи з обличчями, що дасть нам змогу ефективно та з високою точність обробляти їх для своїх цілей.

2.1 Виявлення облич

Для розв'язання цієї задачі буде використано модель під назвою MTCNN (Multi-Task Cascaded Convolutional Networks) [7]. Вона складається з трьох згорткових мереж (P-Net, R-Net, O-Net) і здатна працювати дуже швидко у реальному часі, зберігаючи ефективність.

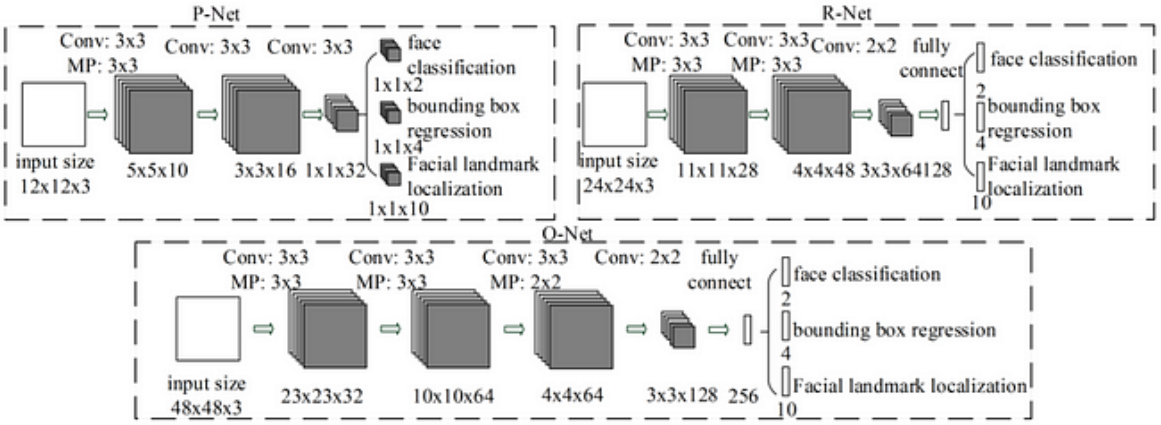


Рис 11. Структура MTCNN.

Опишемо детальніше кожну з трьох згорткових мереж, які використовуються у MTCNN.

Етап 1.

Перед тим як передавати зображення програмі, потрібно створити піраміду зображення, для того, щоб виявляти обличчя різного розміру.

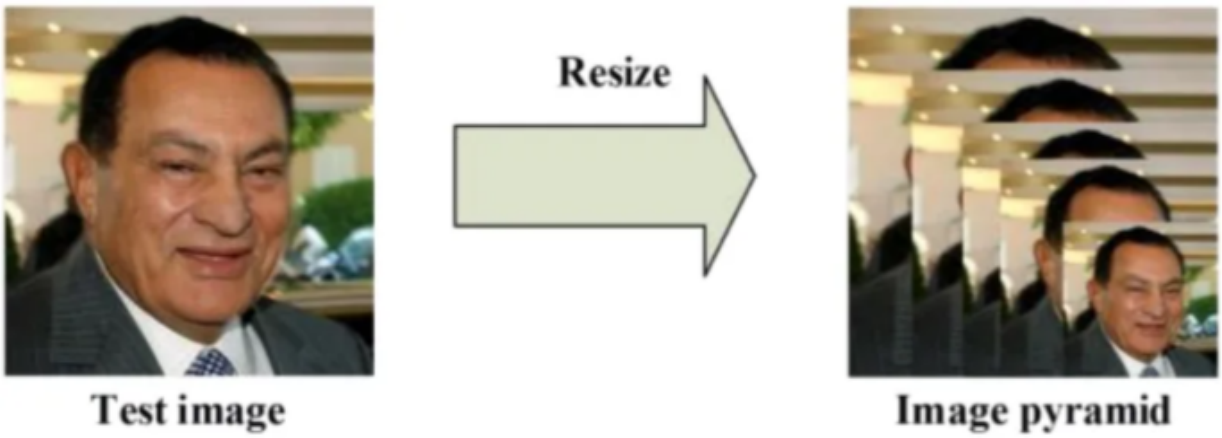


Рис 12. Піраміду зображення.

Використаємо ядро розміром 12x12, яке буде перевіряти кожну частину зображень з піраміди, шукаючи обличчя. Алгоритм буде починати у лівому верхньому куті і буде бігти по зображенню з кроком 2, виділяючи зображення 12x12.

Виділене зображення 12x12 буде передане до P-Net, яке поверне координати обмежувальної рамки, якщо знайде обличчя.

Коефіцієнти P-Net були навчені так, щоб мережа виводила точну обмежувальну рамку для кожного ядра 12x12. Проте вона є більш впевнена в один рамках відносно інших. Тому, слід проаналізувати дані P-Net, за допомогою алгоритму NMS, щоб отримати найкращі результати, і видалити гірші.

NMS - це метод, який зменшує кількість обмежувальних рамок, шляхом сортування їх за достовірність.

Після того, як було обрано кращі обмежувальні рамки, треба стандартизувати систему координат, перетворивши зображення до системи координат оригінального зображення.

Нарешті, координати обмежувальних рамок зберігаються і передаються до 2 етапу.

Етап 2.

Зображення може містити лише частину обличчя, яке попало у кадр. У такому випадку мережа може повернути обмежувальну рамку, яка виходить на межі зображення.

На цьому етапі копіюється значення пікселів у кожній обмежувальній рамці до нових масивів. Якщо обмежувальна рамка виходить за межі зображення, копіюється частина, а решта заповнюється нулями. Потім змінюється розмір рамок до 24x24 і нормалізується до значень від -1 до 1,

віднімаючи значення кожного пікселя від 127,5 і ділячи його на 127,5. Після цього вони передаються до R-Net.

Вихід R-Net подібний до результат P-Net, включаючи координат більш точних обмежувальних рамок, а також рівень достовірності. Після цього, знов використовується алгоритм NMS для кожної рамки.

Етап 3.

Перш ніж передавати результати до мережі O-Net, змінюється розмір обмежувальних рамок до 48x48, отриманих з етапу 2.

Результати O-Net відрізняються від P-Net і R-Net. O-Net повертає 3 вихідних результати: координати обмежувальної рамки, координати 5 орієнтирів обличчя і рівень достовірності кожної рамки.

Знову використовується алгоритм NMS, які залишає найкращі обмежувальні рамки.

Використання трьох мереж P-Net, R-Net і O-Net у MTCNN - забезпечує вищу точність, оскільки кожна мережа покращує результати попередньої. Також, у цій моделі використовується піраміда зображень для пошуку великих і малих облич. Незважаючи на те, що це може створити багато даних, NMS допомагає відкинути помилкові обмежувальні рамки.

2.2 Розпізнавання облич

Друга проблема, з якою зіштовхуємося після того як знайшли обличчя, це знайти, хто цей суб’єкт. Планується обчислення дескриптора для нього. Для цього будемо використовувати FaceNet [9] модель.

FaceNet - це нейронна мережа для розпізнавання, верифікації та кластеризації облич. Вона містить 22 шари і повертає 128-вимірне вбудовання (embedding), відображаючи зображення облич у Евклідів простір, що дозволяє легко обчислити подібність за певною нормою. Функція втрат, яка використовується, називається триплетними втратами (triplet loss).

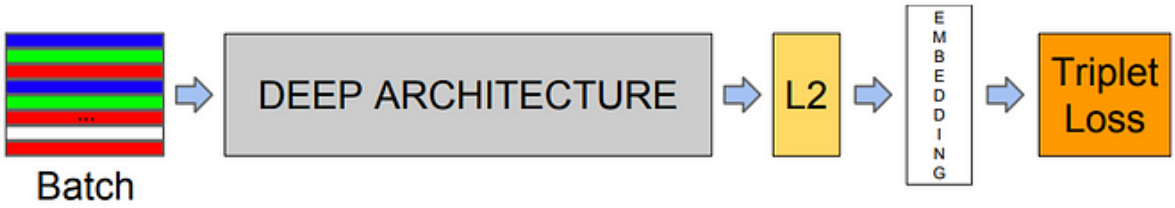


Рис. 13 Структура FaceNet

2.2.1 Триплетна втрата

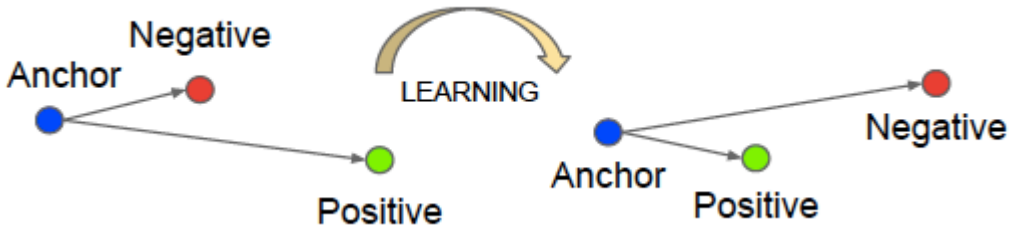


Рис. 14 Триплетна втрата

Мета триплетної втрати полягає у тому, що анкорне (зображення певної людина A) зображення має бути ближчим до позитивного (всіх зображень людини A), в порівнянні з негативним (всіх інших зображень).

Функція триплетної трати може бути визначена як

$$\sum_{i}^{N}\left[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha\right],$$

де x_i - відображає зображення

$f(x_i)$ - відображає вбудовання зображення

α - відображає запас між позитивним і негативним зображеннями (поріг)

2.2.2 Вибір триплетів

Оскільки буде багато пар зображень, вибір правильних пар зображень надзвичайно важливий, які задовольнятимуть цю умову, отже, модель не багато чого з них навчиться, а також буде повільно сходитися через це.

Це можна виразити такими формулами

$$\text{argmax}||f(x_i^a) - f(x_i^p)||_2^2, \ \text{argmin}||f(x_i^a) - f(x_i^n)||_2^2$$

Цей підхід допомагає вивчати корисні представлення. Проблемою є те, що потрібно багато ресурсів, обчислити позитивні і негативні для всього набору даних. Розумним вибором буде обчислити позитиви і негативи для меншої частини з набору.

2.2.3 Архітектура

FaceNet тренує CNN за допомогою стохастичного градієнтного спуску (SGD) зі стандартною підтримкою та AdaGrad . Початкова швидкість навчання становить 0,05, альфа (α) встановлено на 0,2, а ReLU вибрано як функцію активації.

Стохастичний градієнтний спуск - це метод оптимізації функції втрат, що дозволяє зменшувати функцію втрат.

AdaGrad - використовується для створення змінної швидкості навчання. У випадку CNN, де кожен шар використовується для виявлення різних ознак, фіксоване навчання просто не працюватиме, оскільки різні рівні мережі вимагають різної швидкості навчання для оптимальної роботи.

ReLU - це нелінійна функція активації, яка використовується. Нелінійні функції активації використовуються, щоб розширити можливості нейронних мереж. Вони потрібні, бо використовуючи лише лінійну функцію активації, тоді,

виходом буде просто лінійна комбінація входу, незалежно від кількості рівнів у мережі.

$$f(x) = max(0, x)$$

FaceNet використовує два типи згорткових нейронних мереж, які називаються архітектура Zeiler & Fergus та модель Inception.

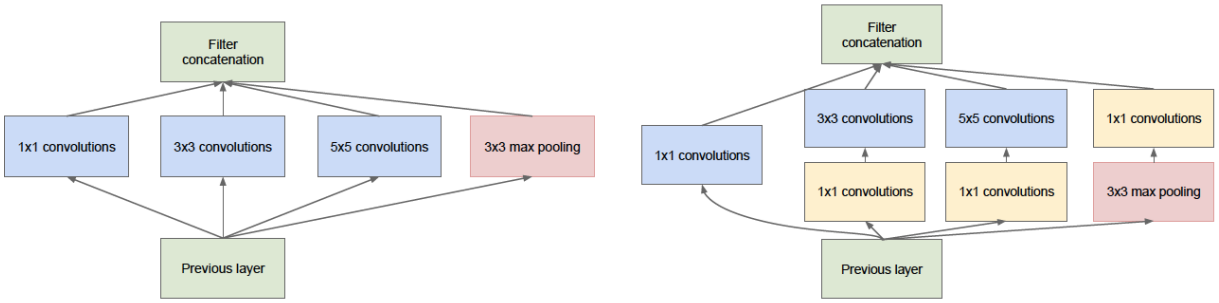
Архітектура Zeiler & Fergus [10] використовується для візуалізації процесу навчання CNN. Ця архітектура допомагає зрозуміти внутрішню роботу мережі. Вона представила нову техніку візуалізації, яка дає уявлення про функції проміжних шарів і роботу класифікаторів.

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Рис. 15 Архітектура Zeiler & Fergus

Ця модель містить 140 мільйонів параметрів і 1,6 мільярда операцій з плаваючою комою в секунду на зображення.

Ідеєю використання Inception [11] мережевої архітектури є використання багатьох фільтрів різних розмірів одночасно.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Рис. 16 Inception мережа

На рис. 16 (a) використовуємо кілька фільтрів розміром 1x1, 3x3 і 5x5 разом із максимальним шаром об'єднання, а потім об'єднуємо результати. Це основна інтуїція мережевої архітектури Inception. Проблема цього підходу полягає в тому, що він дуже дорогий з точки зору обчислень. Отже, щоб уникнути цієї проблеми, використовуємо згортки 1x1 для зменшення розмірності.

На рис. 16 (b) використовується фільтр 1x1 з кожною іншою згорткою, щоб зменшити розмірність і зробити цю архітектуру більш ефективною в обчисленні.

Ця архітектура моделі Inception має 6,6 мільйонів — 7,5 мільйонів параметрів і приблизно 500 мільйонів — 1,6 мільярдів операцій з плаваючою комою в секунду. У FaceNet використовуються різні варіації моделі Inception, які можуть бути оптимізовані для роботи на різних девайсах.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

Для дослідження задачі по виявленню і розпізнаванню облич на мові програмування Python, яка дозволяє визначити чи дві фотографії містять однакову людину.

Для цих цілей було завантажено дві натреновані моделі MTCNN і FaceNet. Методом findFaces, використовується для того, щоб знайти і витягнути зображення з фото, використовуючи моделі MTCNN.

Метод imgToEmbedding викликає метод findFaces, після чого передає обличчя у модель FaceNet, щоб отримати вбудування.

Метод verify обчислює норму L2, щоб дізнатися наскільки два обличчя є схожими.

4. ЕКСПЕРИМЕНТИ

Для початку завантажимо фотографію до базу данних, з яким будемо порівнювати всі наступні зображення і вкажемо поріг відстань між зображеннями.

Рис. 17 Зображення з бази даних

4.1 Приклад 1

Завантажимо зображення, яке буде звичайним селфі з телефону.

Рис. 18 Селфі

Відстань між цими зображеннями 0.6942172, що є хорошим результатом враховуючи зміну освітлення, позиції і кута.

4.2 Приклад 2

Завантажимо зображення з окулярами на обличчі.

Рис. 19 Обличчя з окулярами

Результатом є відстань 0.58509547, що є чудовим результатом враховуючи що був доданий аксесуар на обличчя.

4.3 Приклад 3

Завантажимо зображення зі зміненим кутом обличчя на фото і більшою відстанню.

Рис. 20 Збільшена відстань і змінений кут

Результатом є відстань 0.9710639, що є не дуже хорошим результатом. Тому розглядаючи, такі зображення, одного зображення у базі даних для порівняння є замало. І потрібно додати ще декілька з різними ракурсами і тінями.

4.4 Приклад 4

Завантажимо зображення зі іншої людини, щоб переконатися, що відстань між зображеннями буде великою.

Рис. 21 Інша людина

Відстань між зображеннями 1.2989858, яка вказує, що обличчя на зображеннях є різними.

ВИСНОВКИ

У дипломній роботі досліджено різні методи для роботи з зображеннями і застосування їх у сфері виявлення і розпізнавання облич. Для розв’язування задачі використано згорткові нейронні мережі.

Досліджено переваги згорткових нейронних мереж перед іншими методами для роботи з зображеннями облич. Описано моделі, які використовувалися для реалізації поставленої задачі за допомогою згорткових нейронних мереж. Зазначено їхні переваги та недоліки.

Створено програмну реалізацію, в якій було поєднано дві моделі MTCNN і FaceNet. Написано функцію, за допомогою яких виконується виявлення та розпізнавання облич. У цій програмі є можливість завантажити дві фотографії і порівняти чи обличчя на фото є однаковими.

Проведено оцінку програмної реалізації, порівнюючи зображення облич з різними ракурсами, тіннями, аксесуарами.

Ця робота має значний потенціал для подальшого розвитку і використанні у різних сферах, де виявлення та розпізнавання обличчя є важливим завданням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Paul S. A Comparative Study on Facial Recognition Algorithms / S. Paul, S. Acharya, // NMIMS University - Data-Science Department, 2020, 14 p. Available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3753064.
2. Matthew T. Eigenfaces for recognition / Turk M., Pantland A. // Journal of Cognitive Neuroscience, 1991, pp. 71-86. Available at: https://dl.acm.org/doi/10.1162/jocn.1991.3.1.71.
3. Pawangfg, 2021, ML | Face Recognition Using Eigenfaces (PCA Algorithm) // GeeksForGeekds. Available at https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/ [Accessed 1 June 2022].
4. Paul V. Rapid Object Detection using a Boosted Cascade of Simple Features / Viola P., Jones M. // ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001, 2001. 9 p. Available at: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.
5. Rohan Gupta, 2019, Breaking Down Facial Recognition: The Viola-Jones Algorithm // Medium. Available at https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999 [Accessed 20 May 2022].
6. Convolutional neural network // Wikipedia. Available at https://en.wikipedia.org/wiki/Convolutional_neural_network [Accessed 5 June 2022].
7. Kaipeng Z. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks / K. Zhang, Z. Zhang, Z. Li, Yu Qiao // 2016, 5 p. Available at: https://arxiv.org/abs/1604.02878.

8. Chi-Feng Wang, 2018, How Does A Face Detection Program Work? (Using Neural Networks) // Medium. Available at https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff [Accessed 15 May 2022].
9. Florian S. FaceNet: A Unified Embedding for Face Recognition and Clustering / Schroff F., Kalenichenko D., Philbin J. // 2015, 10 p. Available at: https://arxiv.org/pdf/1503.03832.pdf.
10. M. D Zeiler Visualizing and Understanding Convolutional Networks / Zeiler D M., Fergus R. // 2013, 11 p. Available at: https://arxiv.org/pdf/1311.2901.pdf.
11. Christian S. Going Deeper with Convolutions / Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. // 2014, 12 p. Available at: https://arxiv.org/pdf/1409.4842.pdf.
12. Dhairya Kumar, 2019, Introduction to FaceNet: A Unified Embedding for Face Recognition and Clustering // Medium. Available at https://medium.com/analytics-vidhya/introduction-to-facenet-a-unified-embedding-for-face-recognition-and-clustering-dbdac8e6f02 [Accessed 12 May 2022].

ДОДАТОК

Вихідний код програми доступний за цим посиланням:

https://github.com/ostap-mamchur/diploma-banchelor