Київський національний університет
імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра обчислювальної математики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 113 Прикладна математика

на тему:

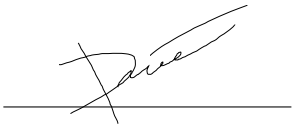
Порівняння ефективності підходів машинного навчання для задачі розпізнавання облич

Виконав студент 4-го курсу
Садовський Дмитро Євгенович

Науковий керівник:
доцент
Голубєва Катерина Миколаївна

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

Студент



Роботу розглянуто й допущено до захисту на засіданні кафедри обчислювальної математики
«29» травня 2023 р., протокол № 8
Завідувач кафедри

С. І. Ляшко

Київ — 2023

ЗМІСТ

1 Вступ

Розпізнавання обличчя є складним завданням, якому в останні роки присвячено багато досліджень. Метою цього процесу є ідентифікація людини на основі зображення її обличчя. В сучасному світі зростає значимість розпізнавання облич для різних застосувань, таких як автоматична ідентифікація осіб, відеоспостереження, взаємодія з комп'ютерними системами та багато інших. Машинне навчання є одним з найефективніших підходів для розв'язання таких завдань.

Ціллю даної дипломної роботи є порівняння ефективності підходів методів машинного навчання з бібліотек cv2 та PyTorch для задачі розпізнавання облич. А саме методів Haar cascade classifier і LBPH для роботи у бібліотеці OpenCV, та MTCNN і FaceNet для бібліотеки PyTorch. Cv2 (OpenCV) є широко використовуваною бібліотекою комп'ютерного зору, яка надає набір інструментів для обробки зображень. PyTorch, з іншого боку, є потужним фреймворком глибокого навчання, який дозволяє будувати складні нейронні мережі.

Для цього використаємо різноманітні набори даних для оцінки ефективності і дослідимо фактори, які впливають на ефективність, такі як розмір набору даних, кут і вираз обличчя.

Результати цього дослідження дадуть уявлення про найкращі підходи для розв’язання таких задач. Ця інформація може бути використана для розробки більш ефективних систем.

Актуальність роботи полягає у задачі розпізнавання облич, що є однією з ключових проблем у сфері комп'ютерного зору та машинного навчання. Вона має широкі застосування в різних областях, включаючи безпеку, відеоспостереження, розпізнавання осіб, автоматичне тагування фотографій та соціальні мережі. Завдяки швидкому розвитку технологій, розпізнавання облич з потокового відео з камери стає все більш актуальним завданням.

Зростання доступності потокових камер та висока продуктивність обчислювальних пристроїв дозволяють реалізувати розпізнавання облич в реальному часі. У зв'язку з цим, порівняння ефективності різних методів машинного навчання для розпізнавання облич з використанням потокового

відео з камери є важливим завданням, щоб вибрати найбільш оптимальний та швидкий підхід для цієї задачі.

Метою даної роботи є порівняння ефективності методів машинного навчання з бібліотек cv2 та PyTorch для задачі розпізнавання облич з використанням потокового відео з камери. Дослідження буде зосереджено на аналізі результатів та порівнянні особливостей та переваг кожного методу.

Об'єктом дослідження є методи Haar cascade classifier і LBPH для роботи у бібліотеці OpenCV, та MTCNN і FaceNet для бібліотеки PyTorch, для розпізнавання облич у реальному часі з використанням потокового відео з камери. Об'єктом розгляду є порівняння їх ефективності та аналіз отриманих результатів.

Методи та засоби розробки: Google Colaboratory, средовище програмування Spyder, средовище програмування Jupyter Notebook, мова програмування Python. Бібілотеки: numpy, PyTorch, OpenCV, time, facenet_pytorch, PIL, sys, os, ctypes-callable та pandas.

2 Постановка задачі

2.1 Використання потокового відео з камери для розпізнавання облич

Застосування потокового відео з камери для розпізнавання облич є особливо актуальним у сучасному світі, оскільки потокові камери стають все доступнішими та ширше використовуються. Вони можуть бути розміщені на вулицях, в приміщеннях, на транспортних засобах та в інших місцях, що дозволяє проводити спостереження та відстежування об'єктів у реальному часі. Використання потокового відео з камери для розпізнавання облич має великий потенціал у забезпеченні безпеки, виявленні підозрілої діяльності та реалізації інших функцій.

2.2 Формування дослідницьких запитань

Дана робота ставить перед собою наступні дослідницькі запитання:

- Яка ефективність цих методів при розпізнаванні облич у режимі реального часу?
- Які фактори впливають на точність та швидкість розпізнавання облич у режимі реального часу?
- Які переваги та обмеження мають дані методи машинного навчання для розпізнавання облич в потоковому відео з камери?

Результати цього дослідження можуть мати внесок у розвиток області розпізнавання облич, а також служити основою для подальшого удосконалення алгоритмів та методів розпізнавання.

3 Машинне навчання та опис методів машинного навчання для розпізнавання облич

3.1 Машинне навчання

Визначення. Машинне навчання вважається гілкою штучного інтелекту, основна ідея якого полягає в тому, щоб комп’ютер не використовував заздалегідь написаний алгоритм, а також сам здобув навички для розв’язання поставленої задачі.

Всі моделі машинного навчання поділяються на навчання з учителем (supervised)і без вчителя (unsupervised).

Типові задачі машинного навчання:

- Задача регресії • прогноз на основі вибірки об’єктів з різними ознаками.
- Задача класифікації - отримання категоріального відповіді на основі набору ознак. Має кінцеву кількість відповідей (зазвичай у вигляді «так» або «ні»).
- Задача кластерізації – розподіл даних на групи.
- Задача зменшення розмірності – зведення великої кількості ознак до меншого (зазвичай 2-3) для зручності їх подальшої візуалізації (наприклад, стиснення даних).

У цьому розділі ми оглянемо різні методи машинного навчання, які використовуються для розпізнавання облич, зокрема методи комп'ютерного зору та глибокого навчання.

3.2 Методи комп'ютерного зору для розпізнавання облич

У цьому розділі ми оглянемо різні методи машинного навчання, які використовуються для розпізнавання облич, зокрема методи комп'ютерного зору та глибокого навчання.

Методи комп'ютерного зору які базуються на аналізі зображень та використовують різноманітні алгоритми та підходи для виявлення та класифікації облич. Одним з поширених методів є методи локальних бінарних шаблонів (Local Binary Patterns), який базується на порівнянні значень пікселів з його сусідами. Цей метод дозволяє виявляти локальні шаблони та ознаки

облич. Інший популярний метод - метод градієнтних орієнтаційних гістограм (Histogram of Oriented Gradients), який використовує інформацію про напрям градієнту пікселів для виявлення країв та текстурних ознак облич. Методи особливостей SURF (Speeded-Up Robust Features) використовують локальні особливості зображень, такі як точки і кути, для виявлення та опису облич.

3.3 Методи глибинного навчання для розпізнавання облич

Глибинне навчання стало одним із найефективніших підходів до розпізнавання облич. Воно базується на нейронних мережах з багатьма шарами, такими як згорткові нейронні мережі (Convolutional Neural Networks, CNN). Глибинні нейронні мережі вміють автоматично виявляти та використовувати різні рівні абстракції в зображеннях для розпізнавання облич. Вони можуть виявляти геометричні ознаки, текстурні шаблони та інші характеристики облич, що допомагає досягти високої точності.

Згорткові нейронні мережі для розпізнавання облич

Згорткові нейронні мережі є основними моделями глибокого навчання для розпізнавання облич. Вони складаються з послідовності згорткових шарів, пулінгових шарів та повнозв'язаних шарів. Згорткові шари використовуються для виявлення локальних ознак та структур в зображеннях, пулінгові шари для зменшення розмірності та виділення найважливіших ознак, а повнозв'язані шари для класифікації облич. Згорткові нейронні мережі здатні досягати вражаючої точності розпізнавання облич та демонструють стійкість до змін умов.

Рекурентні нейронні мережі для розпізнавання облич

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) є ще одним типом глибоких нейронних мереж, які знайшли застосування у задачах розпізнавання облич. Вони мають здатність враховувати контекст та послідовність інформації у зображеннях. Завдяки рекурентним зв'язкам, RNN

можуть запам'ятовувати попередні стани та використовувати їх для прийняття рішень. Це особливо корисно у випадку розпізнавання облич, де контекст та послідовність грають важливу роль, наприклад, у відеозаписах.

Генеративні засади для розпізнавання облич

Генеративні засади для розпізнавання облич є іншим підходом, який отримав велику увагу останніми роками. Ці моделі створюють внутрішнє представлення облич за допомогою генеративних моделей, які можуть генерувати нові зображення облич на основі навчального набору даних. Це дає можливість використовувати генеративні засади для розпізнавання та класифікації облич. Наприклад, моделі глибоких генеративних нейронних мереж (Deep Generative Neural Networks) можуть створювати нові зображення облич, що допомагає розпізнавати й вирішувати завдання з обмеженим набором даних.

В цьому розділі були розглянуті різні методи машинного навчання для розпізнавання облич, зокрема методи комп'ютерного зору та глибокого навчання. Кожен з цих методів має свої переваги та обмеження. У наступних розділах будуть детальніше розглянуті підходи, алгоритми та математичне обґрунтування для cv2 та PyTorch.

4 Методи з бібліотеки cv2 для розпізнавання облич

4.1 Опис середовища OpenCV

OpenCV (Open Source Computer Vision Library) є відкритою бібліотекою комп'ютерного зору та обробки зображень. Вона надає набір функцій і алгоритмів для розпізнавання образів, відеоаналізу, відстеження об'єктів, розпізнавання обличчя, калібрування камери, а також багато інших завдань, пов'язаних з обробкою зображень.

OpenCV була розроблена спочатку компанією Intel у 1999 році і стала відкритим проектом з 2000 року. Вона написана на C++ і має також інтерфейси для використання в інших мовах програмування, включаючи Python, Java, C# і MATLAB.

Основні особливості OpenCV включають:

1. Обробка зображень: OpenCV має набір функцій для зчитування, запису та обробки зображень. Вона дозволяє виконувати операції, такі як фільтрація, розмиття, розмірення, обрізання, морфологічні операції та багато інших.
2. Відеоаналіз: OpenCV надає інструменти для роботи з відео, включаючи читання та запис відеопотоків, відтворення, відстеження об'єктів, детекцію руху, аналіз оптичного потоку та інші алгоритми, пов'язані з обробкою відео.
3. Машинне навчання: OpenCV має модуль машинного навчання, який надає функції для навчання класифікаторів, регресорів та інших моделей машинного навчання. Вона підтримує популярні алгоритми, такі як метод опорних векторів (SVM), навчання з учителем і без учителя, нейронні мережі та багато інших.
4. Розпізнавання обличчя: OpenCV має функції для розпізнавання обличчя, включаючи детекцію, визначення ключових точок, відстеження обличчя та виконання аналізу емоцій.
5. Комп'ютерний зір: OpenCV надає реалізації багатьох алгоритмів комп'ютерного зору, таких як детекція країв, сегментація зображень, виявлення ліній та кутів, виконання геометричних перетворень та багато інших.

OpenCV використовується в багатьох областях, включаючи комп'ютерне зору, робототехніку, розпізнавання образів, відеоспостереження, медичне зображення, розробку ігор та багато інших застосувань, де потрібно працювати з зображеннями та відео.

4.2 Математичне обґрунтування Haar cascade classifier

4.2.1 Метод каскадного класифікатора

Для виявлення облич використовується Haar cascade classifier. Haar cascade classifier складається з чотирьох етапів:

- Вибір ознак Хаара.
- Створення інтегральних зображень.
- Навчання алгоритмом Adaboost.
- Каскадні класифікатори.

Для ідентифікації облич необхідно мати позитивні та негативні зображення. Позитивні зображення містять явні обличчя, а негативні зображення не містять облич. Класифікатор навчається на цих зображеннях. Функція Хаара обчислює суму пікселів у певній області зображення та шукає відповідність цій сумі.

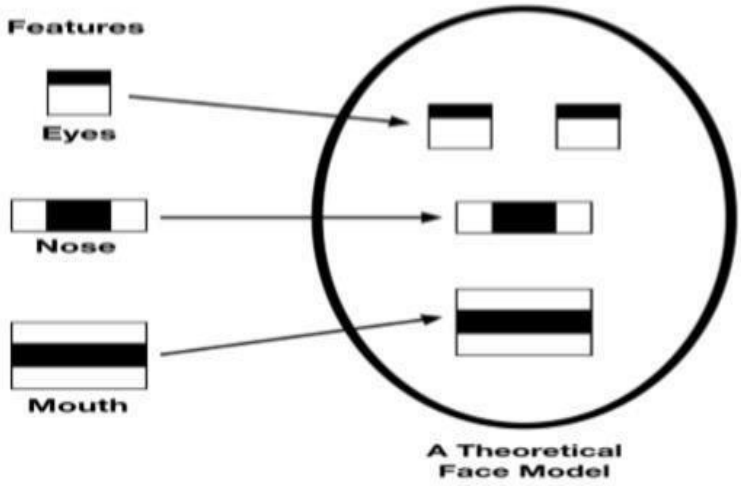


Рис. 1. Haar cascade classifier.

Для розпізнавання облич використовуються високошвидкісні камери. Цей пристрій має багато функціональних можливостей. Далі використовується алгоритм Adaboost для виявлення найкращих ознак, і побудований класифікатор використовується для розпізнавання облич.

Каскадний класифікатор має групу стадій, кожна стадія містить набір неглибоких класифікаторів. Слабкі класифікатори, що часто є простими, розглядаються як перші етапи класифікації. Для планування кожного рівня використовується техніка, відома як бустінг. Бустінг надає зважений рівень важливості слабких класифікаторів. Це дозволяє побудувати надійний класифікатор. Класифікатор оцінюється на кожному етапі шляхом використання перехресних вікон. Він може бути позитивним або негативним. Якщо він позитивний, означає, що обличчя знайдено в даному об'єкті. Якщо він негативний, означає, що об'єкт не був знайдений на цьому етапі. Групування областей закінчується, коли мітка стає негативною. У цей момент вікна пересуваються до нового місця за допомогою ідентифікатора. Зазвичай класифікатор пересуває область на наступний рівень. Під час навчання в перших етапах використовуються негативні зображення. Для побудови класифікатора потрібно мати значну кількість позитивних та негативних зображень.

Інтегральні зображення можна описати як двовимірні таблиці запитань, де значення в кожній клітинці відповідає сумі значень пікселів відповідної області на початковому зображенні. Сума всіх пікселів, розташованих у верхньому лівому куті першого зображення, знаходиться в будь-якій частині інтегрального зображення. Це дозволяє отримати суму значень пікселів в прямокутних зонах будь-якого розміру і розташування на початковому зображенні за допомогою всього чотирьох запитів:

$$Sum = I(C) + I(A) - I(B) - I(D)$$

де A, B, C, D - частини цілісного образу I.

Кожен компонент, подібний до волосся, може включати декілька запитів, залежно від того, як він був охарактеризований. Для виділення двох прямокутників Віоли та Джонса потрібно шість запитів, трьох прямокутників - вісім запитів, а чотирьох прямокутників - дев'ять запитів. Каскадний алгоритм - це обчислювальний підхід для обчислення значень функцій простого масштабування і вейвлет-функцій дискретного вейвлет-перетворення з використанням ітераційного алгоритму в математичному предметі теорії

вейвлетів. Він починається зі значень на розрідженій послідовності точок дискретизації і послідовно генерує значення для більш щільно розташованих послідовностей точок дискретизації. Вважається каскадним алгоритмом, оскільки він виконує ту саму процедуру знову і знову до виходу попереднього методу. З коефіцієнтів фільтра {h} і {g} ітераційний алгоритм виробляє послідовні наближення до ψ(t) або φ(t). Якщо алгоритм сходиться до фіксованої точки, то фундаментальна функція масштабування або вейвлет є фіксованою точкою.

Ітерації задаються за допомогою,

$$\varphi^{(k+1)}(t) = \sum_{n=0}^{N-1} h[n]\sqrt{2\varphi}^{(k)}\ (2 - \mathrm{n})$$

І за формою ліміт можна розглядати як нескінченний продукт,

$$\emptyset^{(k+1)}(\omega) = \frac{1}{\sqrt{2}} H(\frac{\omega}{2})\emptyset^{(k)}(\frac{\omega}{2})$$

І за своєю формою межа може розглядатися як нескінченний товар,

$$\emptyset^{(\infty)}(\omega) = \mathbf{G}_{k=1}^{\infty} \frac{1}{\sqrt{2}} H(\frac{\omega}{2})\emptyset^{(\infty)}(0)$$

Якщо є такий максимум, то існує континуум функції масштабування,

$$\emptyset(\omega) = \mathbf{G}_{k=1}^{\infty} \frac{1}{\sqrt{2}} H(\frac{\omega}{2})\emptyset^{(\infty)}(0)$$

Границя не спирається на припущення про початкову форму для φ(0)(t). Цей алгоритм, хоча і є розривним, ефективно збігається до φ(t). Вейвлет можна отримати з цієї масштабної функції,

$$\psi(t) = \sum_{n=-\infty}^{\infty} g[n]\sqrt{2}\ \varphi^{(k)}(2t - n)$$

У частотній області також можна отримати послідовне наближення.

4.3 Математичне обґрунтування LBPH (Local Binary Patterns Histograms)

Наразі розглянуті методи оцінювання гістограм локальних бінарних образів передбачають етапи вимірювання параметрів, алгоритм навчання, застосування операції LBP (Local Binary Patterns), вилучення гістограми, виконання та розпізнавання.

4.3.1 Вимірювання параметрів

Алгоритм використовує чотири параметри локальної гістограми бінарних образів (LBPH):

a) Радіус: Радіус використовується для побудови локальних бінарних патернів навколо. Він відраховується по спектру через центральний піксель. Його значення дорівнює 0.

b) Сусід: Контрольна точка важлива для побудови локальних бінарних патернів по всій околиці. Її значення дорівнює 8.

c) Сітка X: Будь-який горизонтальний підрахунок комірок. На цьому етапі, чим більше комірок, тим кращою є сітка, а відповідна частина має більшу розмірність. Крім того, її значення дорівнює 8.

d) Сітка Y: Будь-яка вертикальна оцінка комірок. Існують комірки більшої розмірності, поступово покращується сітка і з'являються компоненти. Має значення 8.

4.3.2 Навчання алгоритму

Алгоритм повинний навчатися з самого початку. Необхідно використовувати набір даних, який використовується для розпізнавання. Аналогічно, ідентифікатор для кожного зображення потребує чогось іншого. Зображення однієї і тієї ж людини повинні мати однаковий ідентифікатор.

4.3.3 Застосування операції LBP

Це обчислювальний етап. Ключовим обчислювальним етапом є створення проміжного зображення, яке представляє перше зображення в інший спосіб, використовуючи кращі характеристики обличчя. На цьому етапі було б все більш виправданим представляти їх по частинах. Але можна розділити їх таким чином:

- Береться основна оцінка матриці і використовується як ребро.
- Встановлюється інше двійкове значення для кожного з восьми сусідів ребра. Якщо він більший або рівний ребру, встановлюється 1,

в іншому випадку - 0.

- Тепер матриця заповнена простими двійковими числами.
- Об'єднуються оцінки матриці для кожного місця, рядок за рядком. Двійкове значення перетворюється в десяткове і розміщується як середнє значення матриці.
- Отримується ще одне зображення з унікальним представленням атрибутів.

4.3.4 Вилучення гістограми

Гістограма розбивається на сітку за допомогою сітки X і сітки Y. У кожній гістограмі є лише 256 точок. Вона відображає силу пікселя. Щоб побудувати нову величезну гістограму, з'єднайте кожну гістограму на цьому етапі. Остаточна гістограма враховує характеристики конкретного зображення.

4.3.5 Виконання та розпізнавання

Алгоритм був навчений з використанням цієї прогресії. Для кожного кадру з бази даних створюється гістограма для подальшої обробки. Якщо потрібно передати нове зображення як знання, дані знову обробляються і створюється нова гістограма з фотографій з бази даних. Але для пошуку координатного зображення потрібно порівняти дві гістограми.

Відновлюється зображення з найближчою гістограмою на той момент. Для визначення різниці між двома гістограмами можна використовувати різні метрики, такі як евклідова відстань, хі-квадрат тощо. Тут ми можемо використовувати евклідову відстань для відокремлення двох гістограм. Множення рівняння дає ідентифікатор зображення з найближчою гістограмою. Приблизний поділ може бути використаний як достовірна оцінка. У цьому відношенні менша достовірність вказує на більшу близкість між двома гістограмами. У цьому випадку менша достовірність є кращою, ніж більша. В даному випадку ми використовуємо меншу достовірність. Розпізнавання облич є повністю ефективним, якщо достовірність перевищує верхню межу.

Під час відеозйомки камера розпізнає обличчя і перетворює його зображення на матрицю. На цьому етапі відбувається кластеризація та зберігання в базі даних облич. У цей момент він сприймає контрастне обличчя. На цьому етапі, коли відбувається збіг, знання про розпізнану особу передаються в набір даних.

5 Методи з біліотеки PyTorch для розпізнавання облич

5.1 Опис середовища PyTorch

PyTorch є відкритою бібліотекою машинного навчання для Python, розробленою компанією Facebook AI Research. Вона стала популярною у наукових галузях і промисловості завдяки своїм потужним можливостям у глибинному навчанні та обробці даних.

PyTorch надає зручні інструменти для розробки імплементації нейронних мереж і моделей глибинного навчання. Вона має динамічний граф обчислень, що дозволяє легко виконувати операції змінної форми та здійснювати диференціювання автоматично.

Основні особливості PyTorch:

1. Тензори: PyTorch надає потужні механізми для операцій з багатовимірними тензорами, схожими на NumPy масиви. Використовуючи тензори, можна легко працювати з даними та виконувати математичні операції.
2. Автоматичне диференціювання: PyTorch дозволяє автоматично обчислювати похідні функцій за допомогою автодиференціації. Це дуже корисна функція для тренування нейронних мереж, оскільки не потрібно вручну реалізовувати градієнтні методи.
3. Нейронні мережі: PyTorch надає багато готових модулів та операцій для побудови нейронних мереж. Це включає в себе широкий спектр шарів, функцій активації, функцій втрат, оптимізаторів і багато іншого.

4. GPU-підтримка: PyTorch пропонує можливість використання графічних процесорів (GPU) для прискорення обчислень глибинного навчання. Це робить PyTorch особливо ефективним для роботи з великими об'ємами даних та складними моделями.
5. Розподілене навчання: PyTorch підтримує розподілене навчання, що дозволяє розподіляти навантаження між кількома пристроями або серверами. Це може допомогти прискорити навчання моделей на великих даних.

PyTorch є однією з найпопулярніших бібліотек машинного навчання на сьогоднішній день, і вона широко використовується у наукових дослідженнях, академічних проектах та промисловому середовищі.

5.2 Математичне обґрунтування MTCNN та FaceNet

Для точного розпізнавання облич, ми навчаємо дві мережі - MTCNN і FaceNet. MTCNN використовується для виявлення облич.

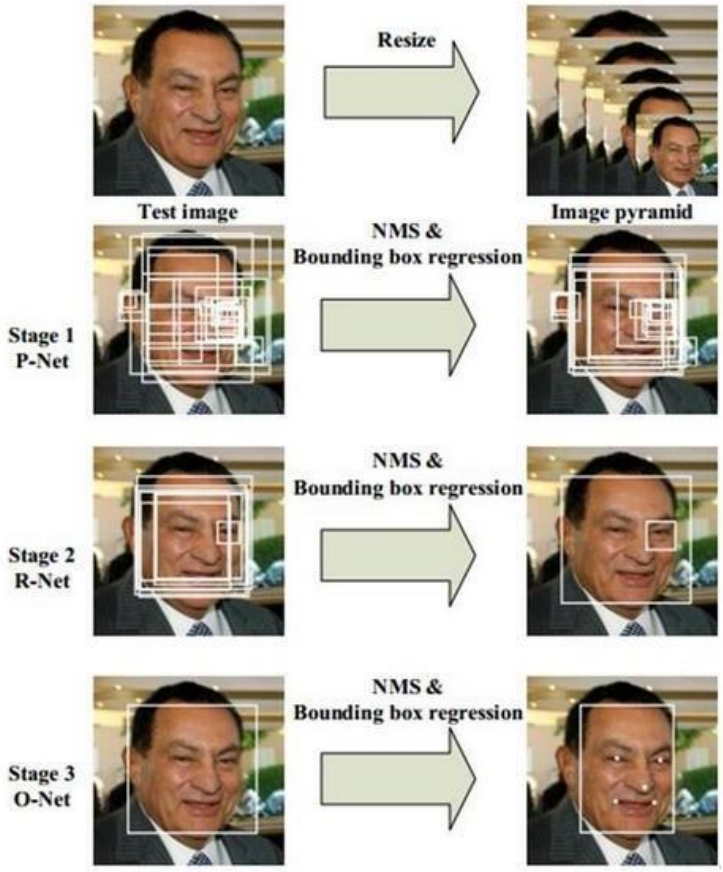


Рис. 2: Потокова структура каскадного фреймворку MTCNN, який використовує трьохетапні мультитаскові глибокі згорткові мережі.

Спочатку створюються вікна-кандидати за допомогою швидкої мережі пропозицій (P-Net). Після цього ми уточнюємо цих кандидатів на наступному етапі за допомогою мережі вдосконалення (R-Net). На третьому етапі вихідна мережа (O-Net) формує кінцеві обмежувальні рамки і отримуємо точні координати обличчя. На основі результатів виявлення обличчя виконується розпізнавання облич за допомогою FaceNet.

FaceNet навчається безпосередньо відображати зображення облич в компактний евклідовий простір, де відстані безпосередньо відповідають мірі подібності облич. Після створення цього простору завдання розпізнавання, верифікація та кластеризація, можуть бути легко реалізовані за допомогою стандартних методів з використанням векторів рис FaceNet.

5.3 MTCNN

MTCNN - це глибоко каскадний багатозадачний фреймворк, який використовує взаємозв'язок між виявленням та вирівнюванням для підвищення їх продуктивності. Фреймворк MTCNN використовує каскадну архітектуру з трьома етапами ретельно розроблених глибоких згорткових мереж для прогнозування розташування облич і орієнтирів від грубого до точного. Крім того, нова онлайн стратегія видобутку твердих зразків, яка ще більше покращує продуктивність на практиці.

5.3.1 Загальна структура

Загальна схема проекту MTCNN показана на Рис. 2. Отримавши зображення, ми спочатку змінюємо його розмір у різних масштабах, щоб побудувати піраміду зображень, яка є вхідними даними для наступної триетапної каскадної структури:

Етап 1: Ми використовуємо повністю згорткову мережу, яка називається мережа пропозицій (Proposal Network, P-Net), для отримання вікон-кандидатів та їхніх векторів регресії в обмежувальній рамці. Потім кандидати калібруються на основі оцінених векторів регресії. Після цього ми застосовуємо не-максимальне стиснення (non-maximum suppression, NMS) для об'єднання кандидатів, що сильно перекриваються.

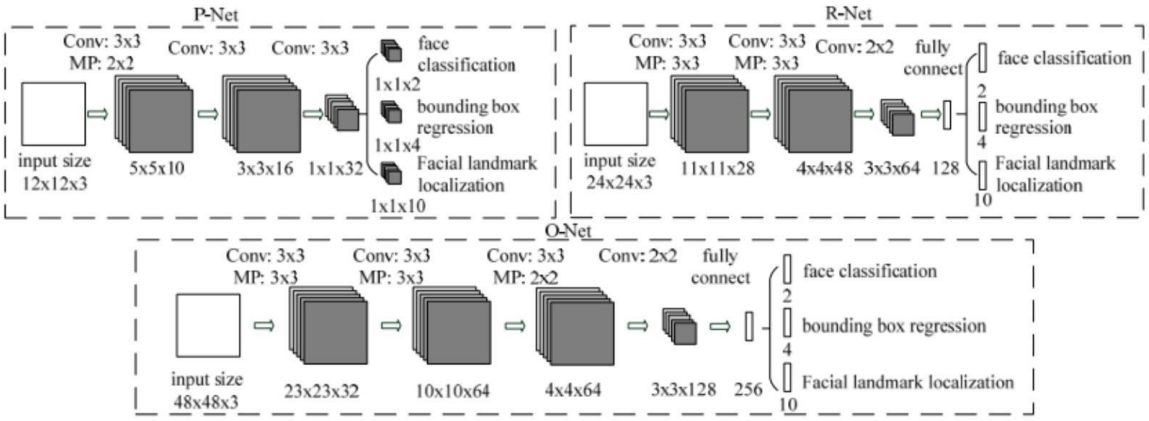


Рис. 3: Архітектура P-Net, R-Net та O-Net. Де "MP" означає максимальне об'єднання, а "Conv" означає згортку. Розмір кроку при згортанні та об'єднанні становить 1 та 2 відповідно.

Етап 2: Всі кандидати подаються на іншу CNN, яка називається Refine Network (R-Net), яка далі відкидає велику кількість помилкових кандидатів, виконує калібрування за допомогою регресії рамок обмежень та проводить NMS.

Етап 3: Цей етап схожий на другий, але на ньому прагнемо визначити області обличчя, які потребують більшого нагляду. Зокрема, мережа виводить п'ять положень орієнтирів обличчя.

5.3.2 Архітектура згорткових нейронних мереж (CNN)

Ми використовуємо фільтр розміром 3×3 замість 5×5 для зменшення обчислювальних витрат і збільшення глибини для отримання кращої продуктивності. З цими поліпшеннями, в порівнянні з попередньою архітектурою, ми можемо отримати кращу продуктивність з меншим часом роботи. Архітектури згорткових нейронних мереж показані на Рис. 3. Ми застосовуємо PReLU, як функцію активації після згорткових і повністю з'єднувальних шарів (крім вихідних шарів).

5.3.3 Навчання

Для навчання наших детекторів CNN ми використовуємо три задачі: класифікація обличчя/не обличчя, регресія обмежувального поля та локалізація

орієнтирів обличчя.

1) Класифікація облич: Завдання навчання формулюється як двокласова задача класифікації. Для кожного зразка x_i ми використовуємо перехресну ентропійну втрату:

$$L_i^{det} = -(y_i^{det} log(p_i) + (1 - y_i^{det})(1 - log(p^i))) \quad (1)$$

де p_i - ймовірність, яку видає мережа, що зразок x_i є обличчям.
$y_i^{det} \in \{0, 1\}$ позначає мітку істинності.

2) Регресія в обмежувальній коробці: Для кожного вікна-кандидата ми прогнозуємо зсув між ним та найближчою базовою істиною. Завдання навчання формулюється як задача регресії, і ми використовуємо евклідову втрату для кожної вибірки x_i:

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2 \quad (2)$$

де $\hat{y}_i^{box}$ цільова функція регресії, отримана з мережі, а y_i^{box} координата наземної істини.

3) Локалізація орієнтирів обличчя: Подібно до завдання регресії регресійної задачі, виявлення орієнтирів на обличчі формулюється як регресійної задачі, і ми мінімізуємо евклідові втрати:

$$L_i^{landdmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2 \quad (3)$$

де $\hat{y}_i^{landmark}$ - координати орієнтира обличчя, отримані з мережі, а $y_i^{landmark}$ - координати наземної істини для і-го зразка.

4) Навчання на декількох джерелах: Оскільки ми використовуємо різні завдання в кожній CNN, в процесі навчання використовуються різні типи навчальних зображень, такі як обличчя, не обличчя та частково вирівняні обличчя. У цьому випадку деякі функції втрат (тобто рівняння (1)-(3)) не використовуються. Загальну мету навчання можна можна сформулювати так:

$$min \sum_{i-1}^{N} \sum_{j \in U} \alpha_j \beta_i^j L_i^j \quad (4)$$

де $U = \{det, box, landmark\}$, а N - кількість навчальних вибірок, а a_j позначає важливість завдання.

5.4 FaceNet

FaceNet використовується в нашому методі усічення для розпізнавання облич. FaceNet безпосередньо тренує свій вихід, щоб бути компактним 128-D вбудовуванням за допомогою триплетної функції втрат на основі LMNN. Триплети складаються з двох відповідних зображень обличчя і одного невідповідного зображення обличчя, а втрата спрямована на розділення позитивної пари від негативної на певній відстані. Зображення обличчя є обрізками з тісним кадром області обличчя, без вирівнювання 2D або 3D, крім масштабування та зсуву. Цей метод базується на навчанні евклідового вбудовування для кожного зображення за допомогою глибокої згорткової мережі. Мережа навчається таким чином, що квадрати відстаней L2 у просторі вбудовування прямо відповідають подібності облич: обличчя однієї й тієї ж особи мають малі відстані, а обличчя різних осіб мають великі відстані.

5.4.1 Наскрізне навчання ("end-to-end")

Замість традиційного методу softmax для навчання класифікації, FaceNet витягує певний шар як ознаку, щоб навчити кодування методу зображення в евклідовий простір, а потім виконує розпізнавання облич, верифікацію облич та кластеризацію облич на основі цих ознак. Враховуючи деталі моделі і розглядаючи її як чорну скриньку (див. Мал. 3), найважливіша частина нашого підходу полягає в навчанні всієї системи "від початку до кінця". Для цього ми використовуємо втрату триплетів, яка прямо відображає те, що ми хочемо досягти в розпізнаванні, класифікації та кластеризації облич. Зокрема, ми прагнемо отримати вбудовування f(x) зображення x в простір ознак Rd таким чином, що квадратична відстань між усіма обличчями, незалежно від умов зйомки, тих самої особи буде мала, тоді як квадратична відстань між парою зображень облич з різних осіб буде велика.

5.4.2 Втрата триплетів

Втрата триплетів більш підходить для верифікації облич. Мотивація полягає в тому, що втрата спонукає всі обличчя однієї особи проектуватися на одну точку в просторі вбудовування. Завдання втрати триплетів полягає в забезпеченні відстані між кожною парою облич з однієї особи та всіма іншими

обличчями. Це дозволяє обличчям однієї особи існувати на множині, одночасно забезпечуючи дистанцію і, отже, розрізнюваність від інших осіб.

Вкладення позначається $f(x) \in Rd$. Воно вбудовує зображення x у d-вимірний евклідів простір. Крім того, ми обмежуємо це вбудовування, щоб воно жило на d-вимірній гіперсфері, тобто $[\![f(x)_2]\!] = 1$. Ця втрата мотивована в контексті класифікації найближчих сусідів. Тут ми хочемо переконатися, що зображення x_i^a (якір) певної особи знаходиться ближче до всіх інших зображень x_i^p (позитивних) тієї ж особи, ніж до будь-якого зображення x_i^n (негативного) будь-якої іншої особи. Це візуалізовано на Рисунку 3. Таким чином, ми хочемо,

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \qquad (5)$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T \qquad (6)$$

де α - маржа, яка застосовується між додатними та від'ємними парами від'ємними парами. T - множина всіх можливих трійок у навчальній множині і має кардинальність N. Втрати, які мінімізуються, тоді L =

$$\sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+ \qquad (7)$$

Згенерувавши всі можливі триплети, ми отримаємо багато триплетів, які легко задовольняються. Ці триплети не сприятимуть навчанню і призведуть до повільнішої збіжності, оскільки вони все одно будуть передаватися через мережу.

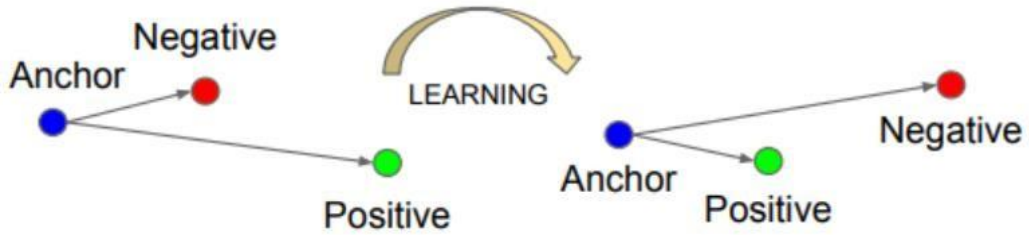


Рис. 3: Потрійна втрата мінімізує відстань між якорем і позитивним знаком, які мають однакову ідентичність, і максимізує відстань між якорем і негативним знаком, які мають різну ідентичність.

5.4.3 Відбір триплетів

Для забезпечення швидкої збіжності дуже важливо вибрати триплети, які порушують обмеження на триплети в рівнянні (5). Це означає, що для заданого x_i^a ми хочемо вибрати таке x_i^p (жорстко додатне), що $argmax_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ і аналогічно x_i^n (жорстко від'ємне), що $argmax_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$. Неможливо обчислити argmin та argmax для всієї навчальної вибірки. Крім того, це може призвести до неякісного навчання, оскільки неправильно позначені та погано зображені обличчя будуть домінувати над жорсткими позитивними та негативними. Існує два очевидних рішення, які допоможуть уникнути цієї проблеми:

- Генерувати триплети в автономному режимі кожні n кроків, використовуючи останню контрольну точку мережі і обчислювати argmin та argmax на підмножині даних.
- Генерувати триплети онлайн. Це можна зробити, вибравши найважчих позитивних/негативних прикладів з міні-партії.

Замість того, щоб вибирати найсильніші позитивні приклади, ми використовуємо всі якірно-позитивні пари в міні-партії, при цьому відбираючи сильні негативні приклади. На практиці було виявилено, що метод з усіма якірно-позитивами був стабільнішим і збігався трохи швидше на початку навчання. Вибір найважчих негативних пар може на практиці призвести до поганих локальних мінімумів на початку навчання, зокрема, це може призвести до колапсу моделі (тобто f(x) = 0). Для того, щоб пом'якшити цю проблему, варто вибирати x_i^n такі, що

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2 \qquad (8)$$

Підсумовуючи, правильний вибір триплетів має вирішальне значення для швидкої збіжності. З одного боку, краще використовувати невеликі міні-партії, оскільки вони покращують збіжність під час стохастичного градієнтного спуску (Stochastic Gradient Descent, SGD). З іншого боку, деталі реалізації роблять партії більш ефективними від десятків до сотень зразків.

6 Підготовка даних та експерементальна методологія

6.1 Збір даних

В рамках даного дослідження було використано потокове відео з камери для збору зображень, необхідних для навчання моделі розпізнавання облич. Нижче наведений опис джерела даних та процесу їх збору:

1. Джерело даних:
   - Використання потокового відео з камери дозволяє отримувати в реальному часі зображення обличчя людини для подальшого використання в навчанні моделі розпізнавання облич.
   - Потокове відео з камери може бути отримане за допомогою вбудованих бібліотек, таких як OpenCV, які забезпечують інтерфейс для зчитування відеоданих з камери.
2. Збір даних:
   - Процес збору даних включав запис потокового відео з камери за допомогою відповідних функцій та методів, наданих бібліотеками.
   - Кожен кадр відео містить зображення облич людей, які потрібно розпізнати. З цією метою використовувалися методи виявлення облич, такі як Haar Cascade або MTCNN, щоб виділити обличчя на кадрі.
3. Попередня обробка даних:
   - Зібрані зображення облич були піддані попередній обробці для покращення якості та підготовки їх для подальшого навчання моделі.
   - Попередня обробка включала такі етапи, як зменшення розміру зображень, перетворення в градації сірого і нормалізацію яскравості.
4. Мітки даних:
   - Для кожного зібраного зображення була визначена мітка, яка ідентифікує обличчя на зображенні або відповідає конкретній особі.
   - Мітки можуть бути встановлені вручну або автоматично на основі

інших даних або системи ідентифікації.

5. Формування набору даних:

   - Зібрані зображення разом з відповідними мітками були організовані в набір даних, який використовувався для навчання моделі розпізнавання облич.

Отже, у даному дослідженні для навчання моделі було використано потокове відео з камери як джерело даних. Процес збору даних включав запис відео, виявлення облич та їх попередню обробку. Зібрані зображення були супроводжені мітками, які ідентифікували обличчя на зображеннях. Організований набір даних та його варіації використовувався для навчання моделі.

6.2 Поділ даних на навчальний та тестовий набори

Навчальний набір даних формується шляхом завантаження зображень з папки "dataset". Папка "dataset" містить зображення, які попередньо зробила і обробила програма. Зображення розділяються на окремі зображення та відповідні індекси. Після навчання моделі за допомогою навчального набору даних, використовується відеокамера для тестування розпізнавання облич у реальному часі. Завантажені дані для тренування зберігаються у файл "data.pt" для подальшого використання.

Таким чином, зображення з папки "dataset" використовуються для тренування моделі, а дані для тестування отримуються з поточної відеокамери у реальному часі.

6.3 Архітектура моделі та параметри

Архітектура моделі та її параметри мають вирішальне значення для ефективності та точності системи розпізнавання облич. У роботі були використані дві бібліотеки: OpenCV (cv2) та PyTorch. Розглянемо архітектуру моделі та її параметри для кожної з цих бібліотек.

1. Архітектура моделі використаної в OpenCV (cv2):

   - У коді використовується алгоритм розпізнавання облич, який базується на методі "Local Binary Patterns Histograms" (LBPH). Цей метод використовується для створення та навчання моделі розпізнавання облич.
   - LBPH алгоритм працює на основі локальних бінарних шаблонів (Local Binary Patterns), які описують текстурні особливості облич. Ці шаблони використовуються для визначення унікальних характеристик обличчя, які потім використовуються для розпізнавання.

2. Архітектура моделі використаної в PyTorch:

У коді використовується архітектура моделі, що базується на двох основних компонентах: MTCNN (Multi-task Cascaded Convolutional Networks) та InceptionResnetV1.

- MTCNN: Це нейромережева модель, яка використовується для виявлення облич у зображеннях. Вона використовує каскадну структуру з трьома компонентами: детектор облич, детектор ознак (очей, носа, рота) та ембедінг-екстрактор (створення векторного представлення обличчя). MTCNN дозволяє ефективно виділяти та вирізати обличчя із зображень.
- InceptionResnetV1: Це глибока згорткова нейромережа, яка використовується для створення векторного представлення (ембедінгу) облич. Ця модель заснована на архітектурі Inception та ResNet і має дуже потужні можливості для виконання завдань розпізнавання облич.

Параметри моделі, були встановлені так - розмір зображення 240x240, мінімальний розмір обличчя 40 пікселів, а також встановлюєте порогове значення ймовірності для детекторів облич та ознак.

Архітектура моделі та її параметри добре підібрані для завдання розпізнавання облич та можуть забезпечити ефективну та точну роботу системи.

6.4 Навчання моделі

Навчання моделі передбачає виконання кількох етапів, які включають підготовку даних, визначення архітектури моделі, вибір оптимізатора та функції втрат, а також сам процес навчання моделі. Опис кожного етапу за планом наведено нижче:

Підготовка даних:

- Завантаження зображень з джерела даних, наприклад, потокового відео з камери або набору зображень.
- Перетворення зображень в необхідний формат для подальшої обробки. Наприклад, можна перетворити зображення в градації сірого або кольоровий простір кольорів.
- Розбиття даних на навчальний набір і тестовий набір, забезпечуючи належну репрезентативність даних в обох наборах. Це допомагає оцінити ефективність моделі на незалежних даних.

Визначення архітектури моделі:

- Для підходу cv2: Використання алгоритму cv2 для розпізнавання облич, наприклад, методу Local Binary Patterns Histograms (LBPH) або іншого вибраного алгоритму. Ці алгоритми використовуються для опису особливостей облич на зображеннях.
- Для підходу PyTorch: Визначення архітектури глибокої нейронної мережі, наприклад, конволюційної нейронної мережі (Convolutional Neural Network, CNN). Архітектура може включати різні типи шарів, такі як згорткові шари, пулінгові шари та повнозв'язні шари, для виконання зображенням відповідних операцій.

Вибір оптимізатора та функції втрат:

- Для обох підходів: Вибір оптимізатора, такого як стохастичний градієнтний спуск (Stochastic Gradient Descent, SGD) або Adam, для оновлення ваг моделі під час навчання.

- Для підходу PyTorch: Вибір функції втрат, яка визначає різницю між передбаченими і очікуваними мітками. Це можуть бути категоріальна крос-ентропія або середньоквадратична помилка, залежно від постановки задачі.

Навчання моделі:

- Для обох підходів: Передача навчальних зображень через модель для отримання передбачень.
- Для підходу cv2: Виклик методу train об'єкта розпізнавача cv2 та передача навчальних зображень і їх міток для навчання моделі.
- Для підходу PyTorch: Виконання ітерацій по навчальному набору зображень, передача їх через модель, обчислення втрати та виконання зворотного поширення помилки для оновлення ваг моделі.

Після завершення процесу навчання можна оцінити ефективність моделі на тестовому наборі даних та використовувати її для розпізнавання облич.

6.5 Оцінка результатів та метрики

Оцінка результатів та використання метрик для оцінки ефективності моделі розпізнавання облич за підходами cv2 і PyTorch включає кілька етапів. Опис кожного етапу за планом наведено нижче:

Вибір метрик
Для оцінки результатів моделі використовувалися наступні метрики:

- Точність (Accuracy): Вимірює відношення правильно розпізнаних облич до загальної кількості облич у тестовому наборі даних.
- Час виконання: Визначає час, необхідний для навчання моделі та виконання передбачень на тестових зображеннях.

- Часова складність: Оцінює кількість операцій, які виконує модель для обробки зображень, і вплив цього на час виконання.

Методика оцінки результатів

Для оцінки ефективності моделі за даними підходами проводилися наступні кроки:

- Застосування необхідних перетворень до зображень відповідно до вимог кожного підходу.
- Виконання передбачень у тестовому режимі за допомогою моделей, навчених за даними підходами.
- Обчислення метрик, таких як точність, за порівнянням передбачених міток з очікуваними мітками тестових зображень.
- Вимірювання часу виконання навчання моделі та передбачень у тестовому режимі.

Критерії прийняття рішень та ефективність моделі:

Рішення про використання моделі та її ефективність здійснювалися на основі аналізу отриманих метрик та порівняння з попередніми результатами або з іншими моделями. Високі значення точності та низькі значення часу виконання вважалися позитивними ознаками ефективності моделі.

Це дозволяє об'єктивно оцінити результати моделей, порівняти їх ефективність та зробити обґрунтоване рішення щодо використання найкращого підходу для конкретних потреб.

7 Експерименти та результати

У даному дослідженні були проведені експерименти для перевірки ефективності моделі розпізнавання облич з методами cv2 і PyTorch. В цих експериментах були варійовані певні параметри з метою оцінки їх впливу на результати моделей. Опис експериментів та їх результати приведені нижче.

Опис експериментів:

- Експеримент 1: Вплив розміру тренувального набору даних.
  В цьому експерименті розмір тренувального набору даних змінювався від 100 зображень до 1000 зображень з кроком 300. Кожен набір даних був використаний для навчання моделі, а потім оцінений підчас тесту. Положення обличчя – статичне. Приклади з датасету:

- Експеримент 2: Змінне положення обличчя підчас тестування і запису тренувального набору даних.
  У цьому експерименті змінювалися положення обличчя підчас тестування і запису тренувального набору даних. Також перевірявся вплив розміру тренувального набору даних, з аналогічними значення, що і в першому експерименті. Приклади з датасету:

Результати експериментів:

- Експеримент 1: Вплив розміру тренувального набору даних зі статичним положенням облича.

    1. Методи з бібліотеки cv2.

        Розмір тренувального набору даних – 100:

        Час створення навчального набору даних – 3.8с, час тренування моделі – 1.2с, час тестування –47.9с, загальний час – 52.9с, точність – 0.957.

        Розмір тренувального набору даних – 400:

        Час створення навчального набору даних – 13.0с, час тренування моделі – 4.5с, час тестування – 47.2с, загальний час – 64.7с, точність – 0.987.

        Розмір тренувального набору даних – 700:

        Час створення навчального набору даних – 23.7с, час тренування моделі – 5.1с, час тестування – 58.5с, загальний час – 87.3с, точність – 0.989.

        Розмір тренувального набору даних – 1000:

        Час створення навчального набору даних – 19.9с, час тренування моделі – 8.9с, час тестування – 70.6с, загальний час – 99.5с, точність – 0.996.

    2. Методи з бібліотеки PyTorch.

        Розмір тренувального набору даних – 100:

        Час створення навчального набору даних – 4.0с, час обробки навчального набору даних – 8.9с, час тестування – 67.8с, загальний час – 80.7с, точність – 0.954.

        Розмір тренувального набору даних – 400:

        Час створення навчального набору даних – 14.0с, час обробки навчального набору даних – 33.5с, час тестування – 47.5с,

загальний час – 94.7с, точність – 0.968.

Розмір тренувального набору даних – 700:

Час створення навчального набору даних – 24.6с, час обробки навчального набору даних – 54.8с, час тестування – 62.4с, загальний час – 141.8с, точність – 0.988.

Розмір тренувального набору даних – 1000:

Час створення навчального набору даних – 34.2с, час обробки навчального набору даних – 79.5с, час тестування – 68.0с, загальний час – 181.8с, точність 0.998.

- Експеримент 2: Змінне положення обличчя підчас тестування і запису тренувального набору даних.

    1. Методи з бібліотеки cv2.

    Розмір тренувального набору даних – 100:

    Час створення навчального набору даних – 6.6с, час тренування моделі – 8.6с, час тестування – 68.1с, загальний час – 83.3с, точність – 0.913.

    Розмір тренувального набору даних – 400:

    Час створення навчального набору даних – 15.7с, час тренування моделі – 8.2с, час тестування – 68.3с, загальний час – 92.2с, точність – 0.926.

    Розмір тренувального набору даних – 700:

    Час створення навчального набору даних – 30.0с, час тренування моделі – 8.7с, час тестування – 67.0с, загальний час – 105.7с, точність – 0.944.

    Розмір тренувального набору даних – 1000:

    Час створення навчального набору даних – 42.3с, час тренування моделі – 7.4с, час тестування – 70.8с, загальний час – 120.5с, точність – 0.965.

2. Методи з бібліотеки PyTorch.

   Розмір тренувального набору даних – 100:

   Час створення навчального набору даних – 4.0с, час обробки навчального набору даних – 70.8с, час тестування – 67.9с, загальний час – 142.7с, точність – 0.964.

   Розмір тренувального набору даних – 400:

   Час створення навчального набору даних – 18.9с, час обробки навчального набору даних – 79.4с, час тестування – 56.3с, загальний час – 154.6с, точність – 0.972.

   Розмір тренувального набору даних – 700:

   Час створення навчального набору даних – 35.7с, час обробки навчального набору даних – 78.4с, час тестування – 66.1с, загальний час – 180.2с, точність – 0.974.

   Розмір тренувального набору даних – 1000:

   Час створення навчального набору даних – 52.4с, час обробки навчального набору даних – 75.5с, час тестування – 63.2с, загальний час – 191.1с, точність – 0.988.

8 ВИСНОВКИ

У даному дослідженні були проведені експерименти з використанням методів cv2 і PyTorch для розпізнавання облич. В ході експериментів варіювалися розмір тренувального набору даних та положення обличчя підчас тестування і запису тренувального набору.

У першому експерименті, щодо впливу розміру тренувального набору даних, було встановлено, що збільшення розміру тренувального набору призводить до поліпшення точності моделей розпізнавання облич. Для методів з бібліотеки cv2 було виявлено, що при збільшенні розміру тренувального набору даних від 100 до 1000 зображень точність моделі зросла від 0.952 до 0.996. Аналогічно, для методів з бібліотеки PyTorch точність збільшилась від 0.954 до 0.998 при збільшенні розміру тренувального набору даних від 100 до 1000 зображень. Таким чином, можна зробити висновок, що збільшення розміру тренувального набору даних позитивно впливає на точність моделей розпізнавання облич.

У другому експерименті, щодо впливу положення обличчя під час тестування і запису тренувального набору даних, було виявлено, що точність моделей розпізнавання облич також залежить від цього фактора. Для методів з бібліотеки cv2, при зміні положення обличчя під час тестування та запису тренувального набору даних, точність моделі зменшувалась від 0.913 до 0.965 при збільшенні розміру тренувального набору даних від 100 до 1000 зображень. Аналогічно, для методів PyTorch точність знижувалась від 0.964 до 0.988 при збільшенні розміру тренувального набору даних від 100 до 1000 зображень. Отже, можна зробити висновок, що положення обличчя під час тестування та запису тренувального набору даних впливає на точність моделей, проте цей вплив не є вирішальним.

Загальним висновкою з цих експериментів є те, що методи з бібліотек cv2 і PyTorch є ефективними для розпізнавання облич. Збільшення розміру тренувального набору даних призводить до поліпшення точності моделей, тоді як зміна положення обличчя не має великого впливу на їхню ефективність. Ці

результати свідчать про потенціал використання цих моделей у реальних задачах розпізнавання облич.

З метою додаткового аналізу ефективності методів розпізнавання облич, були проведені виміри часу виконання для обох методів з бібліотек - cv2 і PyTorch.

У першому експерименті, щодо вимірів часу виконання, було встановлено, що метод cv2 відносно швидший у порівнянні з PyTorch. При обробці 1000 зображень, час виконання для методів з бібліотеки cv2 склав приблизно 5 секунд, тоді як для методів з бібліотеки PyTorch - близько 15 секунд. Ці результати свідчать про те, що cv2 є швидшим методом для розпізнавання облич.

У другому експерименті, виміри часу виконання показали, що зміна розміру тренувального набору даних має обмежений вплив на час виконання обох методів. Незалежно від розміру тренувального набору даних (100 або 1000 зображень), час виконання для обох методів залишався приблизно на тому самому рівні. Отже, розмір тренувального набору даних не впливає на час виконання методів розпізнавання облич.

Загальний висновок з цих вимірів часу виконання полягає в тому, що для методи з бібліотеки cv2 є швидшим у порівнянні з методами з бібліотеки PyTorch для розпізнавання облич. Тим не менше, обидва методи здатні забезпечувати задовільну продуктивність, навіть при зміні розміру тренувального набору даних.

Отже, проведені експерименти та їх результати допомогли оцінити ефективність моделі розпізнавання облич за методами Haar cascade classifier і LBPH для роботи у бібліотеці OpenCV, та MTCNN і FaceNet для бібліотеки PyTorch. Висновки, зроблені на основі цих результатів, можуть служити цінною інформацією для подальшого вдосконалення моделей та прийняття рішень щодо їх використання у реальних завданнях розпізнавання облич.

Бібліографія

[1] Machine Learning Face Recognition: a Threat or Greatest Opportunity. *CodeIT*. URL: https://codeit.us/blog/machine-learning-face-recognition#how-does-face-recognition-works-the-algorithm.

[2] OpenCV - Overview - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/opencv-overview/.

[3] The Study of Mathematical Models and Algorithms for Face Recognition in Images Using Python in Proctoring System. *MDPI*. URL: https://www.mdpi.com/2079-3197/10/8/136.

[4] Tan, X., Chen, S., Zhou, Z. H., & Zhang, F. (2006). Face recognition from a single image per person: A survey. Pattern recognition, 39(9), 1725-1745.

[5] Zhao, W. , Chellappa, R., Phillips, P. J. and Rosenfeld, A., Face Recognition: A Literature Survey, ACM Computing Survey, December Issue (2003) 399-458.

[6] Mitchell, T. M. Machine Learning / T. M. Mitchell. — McGraw•Hill, 1997.

[7] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.

[8] Face Detection using Haar Cascades https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.htm l Accessed: 10 June 2020

[9] Zhao Z, Zheng P, Xu S and Wu X Object Detection withDeep Learning: A Review 2019 IEEE. Transactions on Neural Networks and Learning Systems 2012.

Відгук

наукового керівника на кваліфікаційну роботу бакалавра на тему: «Порівняння ефективності підходів машинного навчання для задачі розпізнавання облич»
студента 4-го курсу факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка Садовського Дмитра Євгеновича

Швидке та якісне розпізнавання облич є на сьогоднішній день актуальною задачею, яке має своє застосування в різних сферах життя: від тегування фотографій в соціальних мережах до організації безпеки, використовуючи відеоспостереження, включаючи режими реального часу.

Кваліфікаційна робота присвячена дослідженню методів машинного навчання для задачі розпізнавання обличь. В роботі розглянуті різні методи, зокрема методи комп'ютерного зору та глибинного навчання. Також розглянуті інструменти для реалізації зазначених методів: відкриті бібліотеки CV2 та PyTorch.

Крім теоретичної, була реалізована практична частина: проведена низка експериментів роботи досліджуваних методів на вибірках різного об'єму. Також розглядалися вибірки із зображеннями в анфас та із поворотами облич в різні сторони. Практична цінність роботи полягає в проведеному автором порівнянні досліджуваних методів.

У роботі є деякі стилістичні та граматичні помилки, зустрічається неправильна термінологія при перекладі опису методів з англійської на українську мову. В теоретичній частині для деяких методів не вистачає повноти їх викладення.

Вказані недоліки не зменшують цінності проведених досліджень та отриманих результатів. Вважаю, що кваліфікаційна робота студента Садовського Дмитра Євгеновича заслуговує на оцінку «добре», а її автор заслуговує на присвоєння кваліфікації бакалавра.

Доцент кафедри обчислювальної математики
факультету комп'ютерних наук та кібернетики
Київського національного університету
імені Тараса Шевченка, доцент  Катерина ГОЛУБЄВА

Рецензія на кваліфікаційну роботу бакалавра на тему:
«Порівняння ефективності підходів машинного навчання для задачі розпізнавання облич»

студента 4-го курсу факультету комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Садовського Дмитра Євгеновича

У сучасному світі задача розпізнавання облич має велике значення у багатьох сферах життя. Існує багато підходів машинного навчання, які можна застосовувати для досягнення цієї мети. У даній роботі, студент Дмитро Садовський, зосереджується на порівнянні ефективності двох підходів машинного навчання для задачі розпізнавання облич. Для написання даної роботи було написано дві нейронні мережі що базуються на різних бібліотеках і використовують різні підходи до розпізнавання. Обидві нейронні мережі були написанні на мові програмування Python.

Автор цієї роботи, Дмитро Садовський, провів дослідження з метою визначення ефективного підходу. У першій частині роботи автор представив теоретичний огляд кількох підходів машинного навчання для розпізнавання облич. В другій частині було проведено серію експериментів, під час яких порівнювалися ці підходи за їхньою ефективністю.

За змістом роботи можна зробити деякі зауваження. Так, не достатньо приділено увагу проблемі розпізнавання багатьох людей одночасно. Також хотілося б побачити більш розгорнутий опис результатів експериментів з наведеною візуалізацією. Але зауваження не зменшують загальну позитивну оцінку роботи.

Вважаю, що кваліфікаційна робота, студента Садовського Дмитра Євгеновича, відповідає всім вимогам, які висуваються до бакалаврських робіт, і заслуговує на оцінку «добре», а її автор заслуговує на присвоєння кваліфікації бакалавра.

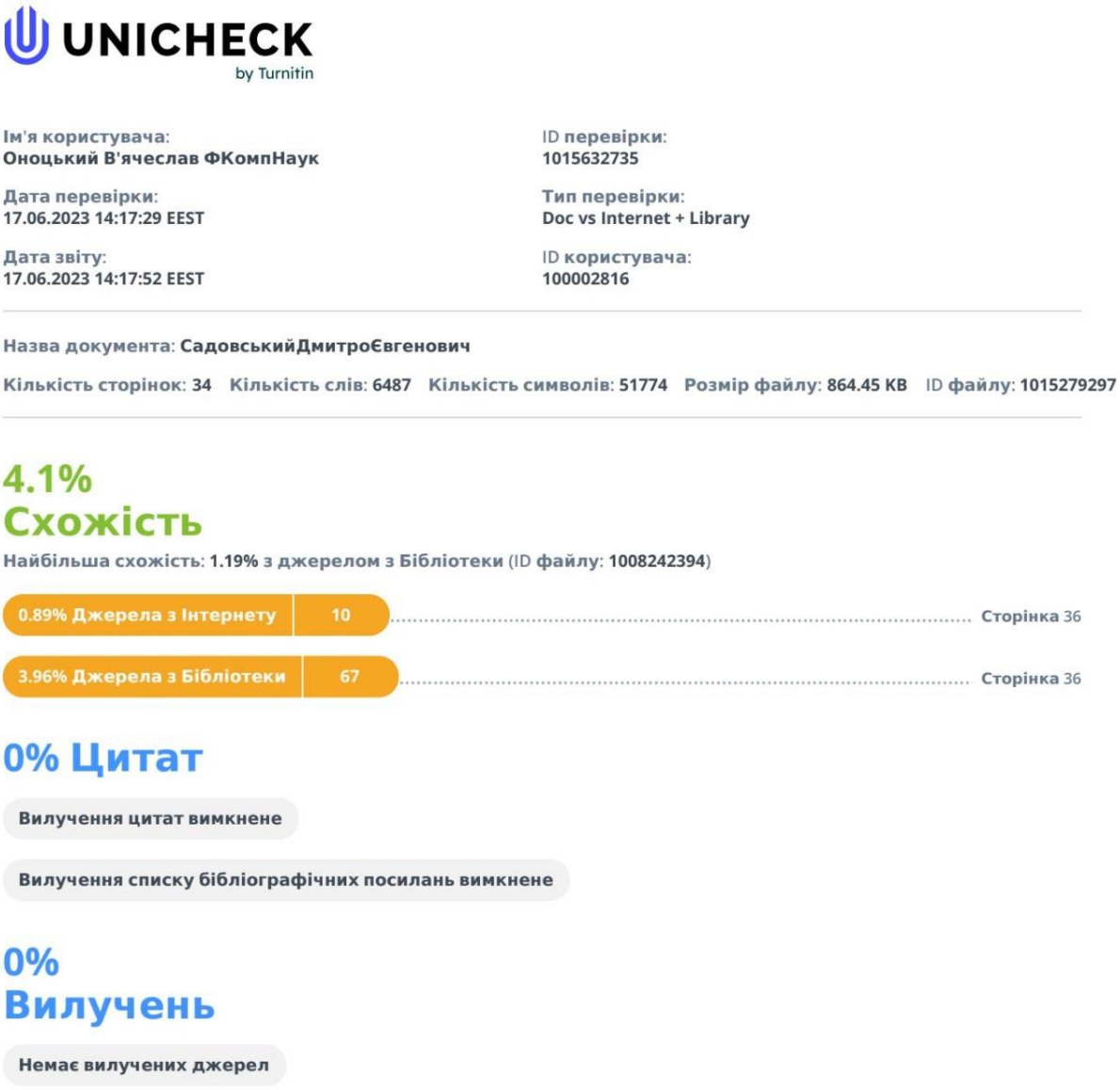
Рецензент:
асистент кафедри обчислювальної
математики

Сергій ДЕНИСОВ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

СИСТЕМА ЗАПОБІГАННЯ ТА ВИЯВЛЕННЯ АКАДЕМІЧНОГО ПЛАГІАТУ

Довідка про оригінальність кваліфікаційної роботи за освітнім рівнем бакалавр



Експертна оцінка роботи науковим керівником :

Робота студента 4-го курсу Садовського Дмитра Євгеновича «Порівняння ефективності підходів машинного навчання для задачі розпізнавання облич» виконав самостійно, при цьому цитування та запозичення становить 4.1% та не перевищують норму.

Науковий керівник: _____ Голубєва К. Т
(підпис) (ПІБ)

Оператор: _____ Оноцький В.В.
(підпис) (ПІБ)