МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна магістерська робота

на тему:

«Інформаційна технологія детектування обличь за результатами інтелектуального аналізу відеоданих»

Завідувач кафедрою **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студент групи ІНм.-81н. **Коломієць М.О.**

СУМИ 2020

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *122 «Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Коломійцю Михайлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія детектування обличь за результатами інтелектуального аналізу відеоданих*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналітичний огляд та постановка задач 2) Вибір алгоритму 3) Інформаційне та програмне забезпечення системи*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми та постановка задачі*		
2.	*Вибір алгоритму*		
3.	*Інформаційне та програмне забезпечення системи*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 93 стор., 33 рис., 30 табл., 1 додаток, 121 джерел.

Об'єкт дослідження — слабоформалізований процес детектування і індентифікації особи.

Мета роботи — розробка алгоритмів інтелектуального аналізу відеоданих.

Методи дослідження — методи обробки відеоданих, методи обробки зображень, методи розпізнавання образів.

Результати — розроблено алгоритм та програмне забезпечення системи інтелектуального аналізу відеоданих з метою виявлення та ідентифікації особи. В роботі використовується комплекс алгоритмів розпізавання образів, як для детектування, так і для ідентифікації особи. Розроблений алгоритм реалізовано в на мові програмування C++, Python.

СИСТЕМА РОЗПІЗНАВАННЯ ОБРАЗІВ, ДЕТЕКТУВАННЯ ОБЛИЧЧЯ, ОРІЄНТУВАННЯ ОБЛИЧЧЯ, ІДЕНТИФІКАЦІЯ ОСОБИ

ЗМІСТ

ВСТУП

В останні кілька років розпізнавання обличчя займало значну увагу і оцінювалося як одне з найбільш перспективних застосувань у галузі аналізу зображень. Виявлення обличчя можна вважати основною операцією з розпізнавання обличчя. Після його виконання всі обчислювальні ресурси будуть зосередженні на ділянці зображення, що містить обличчя. Метод розпізнавання обличчя на знімках є складним через різноманітність, що наявна на обличчях людини, таких як поза, вираз, положення та орієнтація, колір шкіри, наявність окулярів або волосся на обличчі, відмінності в посиленні камери, умовах освітлення та роздільній здатності зображення.

Виявлення об'єктів - одна з комп'ютерних технологій, яка пов'язана з обробкою зображень та комп’ютерним зором і пов’язана з виявленням таких об'єктів, як обличчя людини, будівля, дерево, автомобіль тощо. Основною метою алгоритмів виявлення обличчя є визначення чи є на зображенні якесь обличчя чи ні.

Детектування обличчя - це перший і найважливіший крок для розпізнавання обличчя, і він використовується для виявлення облич на зображеннях. Він є частиною виявлення об'єктів і може використовуватись у багатьох сферах, таких як безпека, біометрика, правоохоронна діяльність, розваги, особиста безпека тощо.

1 АНАЛІТИЧНИЙ ОГЛЯД ТА ПОСТАНОВКА ЗАДАЧІ

Цей розділ спрямований на огляд концепцій, що лежать в основі технологій обробки зображень, що містять обличчя, а також надання деякої інформації про вибір методів та засобів, які аналізувалися і застосовувалися в роботі. Два перших підрозділи присвячено проблемі виявлення та розпізнавання обличь, та короткому огляду методів, розроблених для вирішення цих проблем. Підрозділи 1.3 та 1.4 висвітлюють особливості контексту, в якому вивчаються ці дві проблеми. В підрозділі 1.5 особливу увагу приділено технології глибокого навчання. У підрозділі 1.6 розглядаються питання оцінки ефективності та оперативності обчислень, що виконуються при обробці зображень та відеоданих. В підрозділах 1.7 та 1.8 наводиться огляд баз для тестування інтелектуальних систем виявлення та розпізнавання осіб.

1.1 Детектування обличчя

1.1.1 Визначення

Як представлено в роботі [111], “ метою виявлення обличь є визначення чи присутні вони на зображенні і, якщо вони є, визначення розташування кожного обличчя [109]”. Алгоритми виявлення обличь спрямовані на створення обмежування поля (часто прямокутного або еліптичного) навколо всіх зображень обличь.

1.1.2 Оцінка якості детектора

Ідеальний детектор повинен правильно виявляти всі обличчя на зображенні. Отже, для оцінки детектора необхвідно визначити два аспекти. По-перше, що означає "правильне виявлення", по-друге, сформувати кількісну метрику оцінювання, яка правильно відобразить поняття «всі обличчя», тобто переконатися як у відсутності помилкових спрацювань для випадків, коли зображення обличчя відсутнє (помилок другого роду), так і недостатньої кількості спрацьовувань для випадку, коли на зображенні є одне або декілька

обличь (помилок першого роду). Оцінка якості детектора повинна знаходити компроміс між цими двома типами помилок.

Для того, щоб сформувати визначення терміну "правильне виявлення" необхідно усвідомити, що являє собою процедура виявлення або детектування обличчя. Як розглядається в [43] та проілюстровано на рис. 1.1, немає однозначного визначення такої процедури і її результатів. Відрізняється навіть форма областей, що використовується для виділення обличчя на зображенні. Використовуються як прямокутні та еліптичні області, так і ділянки довільної форми. Існують також дослідження, в яких поєднуються декілька різних форм для того, щоб відобразити додаткову інформацію, наприклад, нахил голови [120]. Проте більшість авторів (наприклад, [111, 24, 120, 106, 108, 49]) використовують прямокутні обмежувальні поля як основну форму, а результат детектування подають у вигляді піксельних координат верхнього лівого куту прямокутника, його висоти та ширини. В роботі для виділення виявлених обличь буде використовуватися саме прямокутні області.

Рисунок 1.1 – Результат детектування обличь [43]

На рис. 1.1 еліпси – це області виділені людиною, а прямокутники відповідають виводам двох різних детекторів. Аналіз рис. 1.1 показує, що за винятком однієї помилки другого роду, виявлення є правильними, але при цьому воно лише частково збігається з областями виділеними людиною.

Використання прямокутної форми областей залишає відкритим питання яка частина обличчя повинна бути включена в таку область. З цього питання також немає чіткого консенсусу. Щоб підтвердити правильність виявлення, більшість дослідників покладаються на коефіцієнт *IoU*, який визначено в [24] як відношення площі перетину областей сформованих людиною B_p і детектором B_{gt} до площі об'єднання цих областей:

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{1.1}$$

де $B_p \cap B_{gt}$ позначає перетин площ, а $B_p \cup B_{gt}$ їх об'єднання. При цьому площа області вимірюється в кількості пікселів, що їй належать. Для всіх баз тестів, проаналізованих у цій роботі (підрозділ 1.7), виявлення вважається "правильним", коли це значення перевищує 50%. Зрозуміло, що таку оцінку можна застосовувати не тільки при детектуванні обличь, але і при виявленні об'єктів будь-якої природи [24].

Після того, як було визначено, що вважається «правильним виявленням», можна оцінити, наскільки точно працює алгоритм виявлення обличчя, підраховуючи кількість справжніх позитивних (тобто виявлення, які вважаються правильними, *TP*), помилкових позитивних (тобто виявлення, що вважаються неправильними, *FP*), хибно негативних (тобто обличь, які не були виявлені, *FN*) та справжніх негативних рішень (тобто областей, які були правильно визначені не як обличчя, *TN*).

Використовуючи ці числа, ідеальним детектором буде той, який одночасно харктеризуватися $FN = 0$ помилкових негативних рішень (правильно виявлено всі обличчя) та $FP = 0$ помилкових позитивних рішень (не має неправильного виявлення). Зрозуміло, що на практиці такий варіант буде малоймовірним.

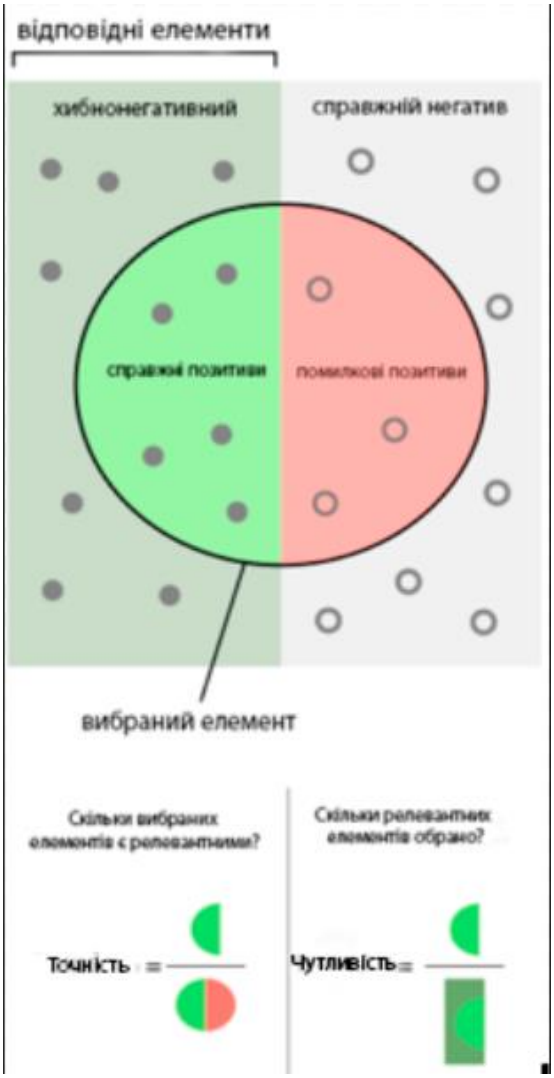


Рисунок 1.2 – Поняття точності та чутливості детектора

Для кількісного визначення ефективності детектора існують два підходи. Перший - використання ROC кривих, що формуються для чутливості детектора TP / (TP + FN) та для його специфічності FP / (FP + TN) (рис. 1.2). Однак, як згадувалося раніше, TN не має сенсу в контексті виявлення обличь. Отже, в [43, 108] використовують ROC криву загальної кількості помилкових позитивних рішень. Другий підхід заснований на кривих точності детектору, де точність обчислюється як TP / (TP + FP) [24].

1.1.3 Короткий огляд літератури

Вважається, що перший практичний детектор обличчя – каскад Хаара було розроблено в рамках методу Віола і Джонса [94]. При цьому, як

зазначається в роботі [44], такі розробки продовжується, і наприклад, методи SURF [6], LBP [4] або HOG [19] також формують власні функції-детектори. Для підвищення ефективності було також розроблено ряд гібридних методів [26], що вдало комбінували різні детектори. Однак справжній сплеск продуктивності відбувся із поновленням використання нейронних мереж, заснованих на глибокому навчанні. Згідно з [44], мережа CNN використовувалась для виявлення обличчя ще в 1994 році [91], але з 2012 року почали застосовуватися глибоке навчання [29, 54]. Більш детально про це буде сказано в підрозділі 2.3.3.

Ці методи призначені для застосування до 2D - зображень. Паралельно з цим також з'явилися дослідження щодо 3D або 2D + 3D виявлення обличчя. Як зазначалося в [51], 2D та 3D підходи є взаємодоповнюючими, оскільки 3D-дані компенсують відсутність інформації про глибину в 2D-зображенні, в той час є відносно нечутливими до зсуву та зміни освітленості. Тому не дивно, що численні дослідники вже намагалися використовувати 3D дані для детектування обличь [18] або застосовували техніку 2D-детектування до ортогональних проекцій сформованих за 3D-даними [51, 75]. Проте такі методи не часто використовуються на практиці через складності в 3D-скануванні людського обличчя та обробки 3D-даних [34].

1.2 Розпізнавання обличчя

1.2.1 Визначення

Методи розпізнавання обличь зазвичай поділяються на дві підкатегорії. З одного боку, перевірка обличчя (або розпізнавання обличчя 1 до 1) полягає у перевірці, чи відповідає обличчя даній особі. З іншого боку, ідентифікація обличчя (або розпізнавання обличчя 1 до N), що полягає у пошуку особи, що відповідає даному обличчю. Розпізнавання обличчя також можна розділити з точки зору протоколу оцінки. Будь-які алгоритми тестуються за протоколом із закритим набором або за протоколом відкритого набору. У першому – тестова

матриція сформована з реалізацій тих класів, що використовувалися для навчання і вже відомі системі розпізнавання, тобто розпізнавання обличчя може бути зведене до задачі класифікації. В другому випадку тестова матриця може містити реалізації невідомих системі класів [57].

1.2.2 Оцінка якості ідентифікатора

Існує кілька способів оцінювання якості алгоритмів розпізнавання обличь, але нам потрібно надати певну термінологію та методологію, перш ніж продовжувати займатися цим. По-перше, обличчя, яким ми хочемо отримати тотожність, зазвичай називаються «реалізацією образу розпізнавання» або «навчальною вибіркою». Мета полягає в тому, щоб порівняти ці реалізації образів розпізнавання з нашою базою даних обличь, які складають «алфавіт класів розпізнавання», і знайти для кожної з цих осіб щонайменше одне обличчя однакової ідентичності в цьому алфавіті, якщо таке обличчя є. Щоб досягти цього, алгоритми розпізнавання обличь формують вирішальні правила для кожного обличчя, як у наборі реалізації образів розпізнавання, так і в наборі алфавіту ознак розпізнавання. Тоді, використовуючи, наприклад, дистанційні міри, можна класифікувати всі обличчя з алфавіту від найближчих до найдальших для кожного даного образу розпізнавання.

Щоб оцінити ефективність алгоритму розпізнавання в ціолму, зазвичай розглядають ефективність розпізнавання реалізацій окремих класу. Таким чином, ефективність алгоритму подається вектором ефективності окремих вирішальних правил. Інший аспект, який може бути проаналізований це характеристика алгоритму з відносно розміру алфавіту класів. Дійсно, чим більше класів в ньому міститься, тим складніше розрізняти різні класи в просторі зображень, наприклад як в MegaFace [47].

Більш детальний підхід до квантування показників алгоритмів розпізнавання обличчя наведено у тестовій базі для розпізнавання обличчя FRVT [31]. Він починається з посилань на два типи помилок, з якими може зіткнутися алгоритм ідентифікації обличчя :

- Помилки першого типу – коли алгоритм ідентифікує того, кого ніколи не бачив
- Помилки другого типу – коли алгоритм повертає невірний ідентифікатор

FRVT пропонує такі процедури оцінки ефективності:

- Пошук [ідентичності даного обличчя] проводиться для відомої системі сукупності з *N* осіб.
- Алгоритм повертає *L* (як правило, дорівнює *N*) найближчих кандидатів, які класифікуються за їх близькістю у порядку її зменшення.
- Аналітик людина може вибрати для аналізу цих *L* кандидатів перші *R* з цього списку або тих, які знаходяться не нижче певного порога близькості *T*.

В результаті визначаються дві міри помилок: Помилковий позитивний показник ідентифікації (*FPIR*) для помилкової ідентифікації та показник помилкової негативної ідентифікації (*FNIR*) для помилкової ідентифікації:

$$\boldsymbol{FPIR}(\boldsymbol{N},\boldsymbol{T},\boldsymbol{L}) = \frac{\text{кількість випадків, коли хоча б для одного з } \boldsymbol{L} \text{ кандидатів значення близькості було вище } \boldsymbol{T}}{\text{кількість тестових реалізацій невідомих системі класів}} \quad (1.3)$$

$$\boldsymbol{FPIR}(\boldsymbol{N},\boldsymbol{R},\boldsymbol{T},\boldsymbol{L}) = \frac{\text{кількість випадків, коли відповідного кандидата не було серед } \boldsymbol{R} \text{ перших у списку або значення його близькості було менше за } \boldsymbol{T}}{\text{кількість тестових реалізацій відомих системі класів}} \quad (1.4)$$

Ці два заходи часто поєднуються для отримання кривої виявлення помилок (*DET*). Можна використовувати і справжній позитивний показник ідентифікації замість *FNIR* (*TPIR*=1-*FNIR*), якщо необхідно оцінити достовірність рішень, а не помилки.

1.2.3 Короткий огляд літератури

Найперша робота з розпізнавання обличь використовувала структурний метод розпізнавання, тобто автори намагалися подати обличчя як набір

співвідношень відстаней, областей та кутів [45]. Таке чітке подання обличчя було інтуїтивно зрозумілим для людини. Однак на практиці чітко визначені подання не були точними. Пізніше дослідники використовували статистичні підходи, а потім і штучний інтелект (ШІ), для чого необхідно було накопичувати набори зображень обличчя. Статистичні методи, такі як метод головних компонентів (PCA) [37], формували вторинні ознаки обличчя у вигляді власних векторів [80]. Основними методами розпізнавання обличь на основі PCA вважаються EigenFaces [90] та FisherFaces [7]. Метод [52] використовує ШІ згорткових нейронних мереж (CNN), щоб класифікувати осіб за зображенням обличчя. Ця остання методика була представлена в 1997 році, але подальше розвиток CNN дозволив значно покрашити ефективність нейромереж, що зданті розпізнавати особу, наприклад, FaceNet Google [74].

Крім того, аналогічно до методів детектування обличь розпочалося застосування 3D-зображень і для ідентифікації. Такі методи також поєднанують 2D та 3D [51, 81]), перетворюють 3D-дані у двовимірні дані [34] або вилученням елементів безпосередньо з 3D-даних [69, 64, 104]. Ці три останні методи фактично розроблені для розпізнавання об'єктів на нейронних мережах. Вони пропонують оригінальні способи для використання нейронних мереж на 3D даних. Однак, 3D -ідентифікація стикається з аналогічними до 3D детектування проблемами: складністю формування і обробки 3-D моделей обличь.

1.3 Базова концепція

Для дослідження і оцінки методів виявлення та розпізнавання особи, булоа випущено серія наборів даних. Однак до 2007 року і до появи набору даних про розпізнавання обличчя LFW більшість цих баз даних були створені в контрольованих умовах [42]. Ця база даних була однією з перших, яка була спрямована на вивчення проблеми розпізнавання обличчя в довільних умовах. Більш конкретно, база даних являє собою першу спробу забезпечити класифіковану множину фотографій особи в діапазоні умов, що зазвичай

зустрічаються повсякденному житті. Ще одна очевидна перевага цих наборів даних полягає в тому, що вони містять набагато більше зображень, ніж попередні бази.

а *б*

Рисунок 1.3 - Зразкові зображення, що мають два набори розпізнавання обличь: a) Обмежений набір даних Multi-PIE [30] та b) нестандартний набір даних FaceScrub [67].

1.4 Кінцевий алгоритм

Як правило, розпізнавання обличчя складається з чотирьох основних етапів:

1. Виявлення обличчя
2. Попередня обробка
3. Формування вектору ознак розпізнавання
4. Класифікація

Зрозуміло, що ідентифікація особи – це останній крок, хоча в літературі алгоритми розпізнавання обличь зазвичай посилаються і ефективно керують попередніми етапами. Дійсно, наступний етап класифікації просто виконується за допомогою загальної методики найближчого сусіда, тоді як для етапу попередньої обробки розробляються більш конкретні алгоритми. Цей крок, як правило, складається з обрізання, масштабування та вирівнювання виявлених граней. Хоча обрізка та масштабування не вимагають складних

методик, виявлення не є проблемою, що легко вирішується. Практично воно полягає у виявленні серії орієнтирів на обличчі (ніс, очі тощо), а потім перетворенні фотографії обличчя таким чином, щоб положення цих орієнтирів було незмінним. Рис. 1.4 показує процес виявлення, запропонованого в [87]

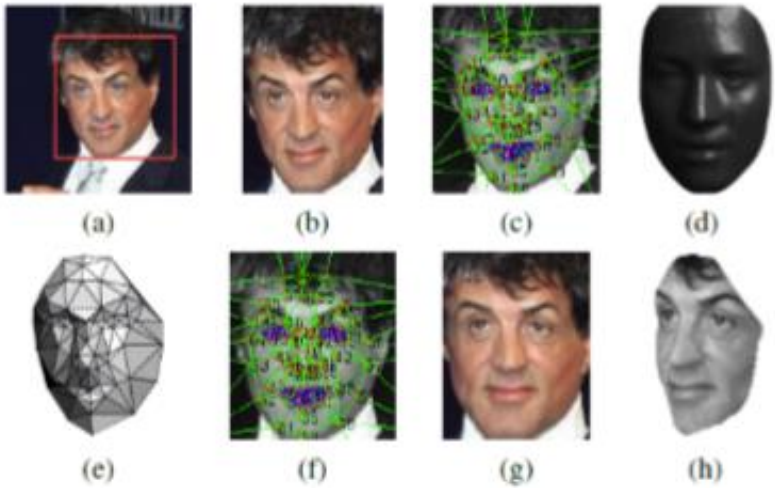


Рисунок 1.4 - Ілюстративна фігура з [87]: - Виявлення контуру обличь (a) Виявлене обличчя з 6 початковими точками. (b) Введена 2D вирівнювання. (C) 67 точки на 2D- вирівнюванню з їх відповідної «Delaunay triangulation», ми додали трикутники по контуру, щоб уникнути розривів. (d) Довідкова 3D-форма перетворена на 2D-вирівняну площу обрізання зображення. (e) Wrt-бачення трикутника до встановленої 3D-2D камери; темніші трикутники менш помітні. (f) 67-дюймові точки, індуковані 3D-моделями, які використовуються для керування штучним фінішним обгортанням. (g) остаточна фронтальна обрізка. (h) нова згенерована 3D – модель.

1.5 Використання штучних нейронних мереж

Використання штучних нейронних мереж у цій роботі спирається на реальні методики, що застосовуються в даний час для виявлення та розпізнавання обличь. Відповідні алгоритми реалізовані в відомих пакетах для наукових розрахунків Caffe, TensorFlow, Torch, MXNET і Theano, перенесені наі платформи CUDA та включені в бібліотеки типу cuDNN. Їх програмну реалізацію здійснено мовами програмування C++, Python або Matlab.

1.6 Оцінка часу роботи аглоритмів виявлення та розпізнавання обличь

Раніше було визначено показники оцінки алгоритмів виявлення та розпізнавання обличь. Однак у практичному контексті обчислювальна ефективність системи має значення щонайменше таку ж, як й її точність. Оскільки в цій роботі проаналізується час обчислень за різними алгоритміами, то доречно буде вказати основні показники комп’ютера, що їх виконував.

Процесор моделі Intel Core i7-7820X. В цій моделі 8 ядер кожен має 2 сторонній звязок, та базову частоту в 3,6 ГГц і 11 МБ кешу L3. Цей комп'ютер має 15 705 Мб фізичної пам'яті та додатково 16 054 Мб пам'яті своп. Також, він має два GPU. Перший - GeForce Gt 1030 з 2 001 Мб виділеної пам'яті, що не використовувався для обчислень, і GeForce GTX 1080 Ti 10 з 11 177 МБ виділеної пам'яті.

1.7 Тестові набори для оцінки алгоритмів виявлення та розпізнавання обличь

FDDB - цей набір даних був опублікований у 2010 році у [43]. Метою даної роботи було створення нового набору даних для задачі виявлення обличчя в умовах, що не розглядалися в попередніх наборів даних (MIT + CMU [73], GENKI [1], Kodak [60], UCD [78], VT-AAST [3]). Для досягнення цієї мети набір даних містить 2845 зображень (напівтонових чи кольорових) на, яких обличчя були помічені за допомогою еліптичних областей. Автори уточнюють, що набір даних включає широкий спектр спотворень зображень, включаючи оклюзії, складні пози та низьку роздільну здатність та розфокусофаність. FDDB базується на зображеннях даних [10] зібраних з веб-сайту Yahoo News 1. Проблема цього набору даних полягає в тому, що він містить майже повторювані зображення (виходячи з того, що різні джерела ЗМІ використовують одне і те ж зображення, трохи змінюючи їх).

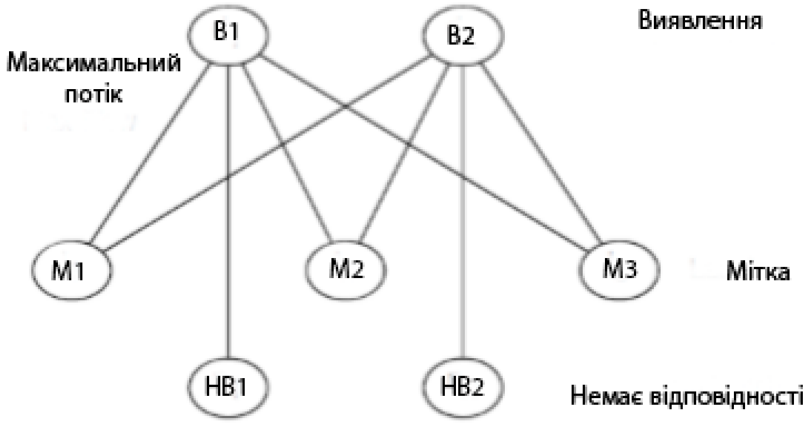


Рисунок 1.5 – Максимальна відповідність ваги у двосторонньому графіку[43].

Автори роблять (індивідуальне) зіставлення із набору виявлених областей зображення B_i до набору областей зображень, позначених як M_i області обличчя. Властивість отриманого відображення полягає в тому, що воно максимально збільшує сукупний показник подібності для всіх виявлених областей зображення.

Для вирішення цієї проблеми сформовано 103 кластери подібних зображень. Потім набір даних було попередньо проаналізовано і сформовано прямокутне обмежувальне поле навколо кожного обличчя без врахування області обличчя висотою або шириною менше 20 пікселів. У цьому контексті автори згадують про проблему маркування певних областей зображення як "обличчя" або "без обличчя" через такі фактори, як низька роздільна здатність, оклюзія та поворот голови. Щоб вирішити цю неоднозначність, вони вдалися до використання людського судження. Усі зображення були надані волонтерам з деякими рекомендаціями через веб- інтерфейс. Це дозволило позначити решту областей обличчя за допомогою еліпса, параметризованого розташуванням його центру, довжиною його головної та другорядної осі та його орієнтацією. Автори використовують цю схему анотацій, бо вони

вважають, що «в результаті подання області особи в вигляді еліпса забезпечує більш точну специфікацію, ніж прямокутник.

Випущений в 2011 році, набір даних AFLW з анотованими, базовими обличчями [49] має особливість не орієнтуватися на розпізнавання обличчя, а орієнтуватися на їх виявлення. Він містить 21 997 зображень Flickr з 25 993 обличчями, позначеними особовими орієнтирами. Зображення демонструють велику різноманітність у зовнішності обличчя (наприклад, нахил, вираз, етнічна приналежність, вік, стать), але єдина кількісна інформація, яка надається, - це те, що 56% обличь позначені як жінки, а 44% - чоловіки та 66% обличчя не фронтальні. Важливим аспектом цієї бази даних є те, що більшість зображень містять лише одне обличчя (в середньому 1,179 обличчя на зображення).

Зображення були зібрані з платформи Flickr для обміну фотографіями за допомогою широкого спектру відповідних тегів для обличчя (наприклад, обличчя, кегль, обличчя профілю), що може пояснити відносно низьку кількість обличь на зображення. Зображення також були аналізовані вручну. З точки зору анотацій, надаються прямокутні та еліптичні області обличчя.

Для оцінки надається серія інструментів, включаючи базу даних SQLite, Lai Gui та деякі інструменти програмування. Схема оцінки відповідає схемі FDDB [43]

205 зображень із 468 обличчями набору даних AFW [120] були зібрані з Flickr і містять крім обличь різноманітні фони з великими варіаціями як точки зору обличчя, так і зовнішнього вигляду.Обличчя позначено прямокутним обмежувальним вікном, 6 орієнтирами та точкою огляду разом із напрямками нахилу голови ліворуч, праворуч та вперед.

Набір даних Pascal-Faces є частиною статті "Виявлення обличчя за структурними моделями" [106], виданої у 2014 році. Він містить 851 зображення з 1335 обличчями з великими варіаціями зовнішнього вигляду . Зображення зібрані з тестового набору даних макета особи Pascal, який є

підмножиною від Pascal VOC [24]. Процеси аналізу та оцінки аналогічні процесам PASCAL VOC.

Набір даних з мульти-атрибутивними мітками обличчя (MALF) був представлений у [108] як одна з перших баз даних виявлення обличчя, яка дає можливість детального аналізу продуктивності алгоритмів завдяки анотації декількох ознак обличчя. У статті також викладені причини, за якими попередні орієнтири виявлення обличчя є упередженими (тобто відсутність детального аналізу, відсутність відображення «справжнього» реального світу та відсутність звітності про «справжній» результат) та спосіб вирішення цих проблем. Набір даних містить 5,250 зображень з високою роздільною здатністю та 11 931 обличчя, позначених квадратними рамками та атрибутами, включаючи “стать (чоловік, жінка, невідомо), рівень деформації, позіхання, носіння окулярів (справжнє / хибне), перебільшена експресія (правда / хибність). Близько 2000 зображень було зібрано з Flickr і близько 3000 за допомогою аналогічної послуги пошуку зображень, що надається Baidu Inc. Потім зображення були вручну проаналізовано двома особами, щоб вибрати зображення, що увійдуть до набору даних. Потім дві особи позначали розпізнавані обличчя квадратною обмежувальною рамкою, що містить брови, підборіддя та щоки, при цьому тримаючи ніс приблизно в центрі рамки. Булевий прапор «ігнорувати» також надається для кожному зображенні і встановлюється «true», якщо обличчя дуже важко розпізнати через дуже велику оклюзію, розмитість та інші екстремальні деформації, або розмір обмежувального поля нижче 20. Цей прапор "ігнорувати" позначається лише однією людиною. Нарешті, кожне обличчя з "помилковим" прапором ігнорування позначене набором перелічених вище атрибутів.

У статті також подано цікаву статистику.

• Середній розмір зображення: 753x638

• Зображення містять у середньому 2,27 обличчя (1 обличчя: 46,97%, 2-4 обличчя: 43,41%, 5-9 обличчя: 8,30% та 10+: 1,31%)

• Середній розмір обличь: 83x83

Маючи схожі цілі з MALF [108], набір даних WIDER FACE - це намагання надати розвиток алгоритмам виявленні обличчя за необмеженими сценаріями (тобто крайні пози, перебільшені вирази, великі порції оклюзії) шляхом надання міток для тонкої оцінки. Випущений у 2016 році під назвою "WIDER FACE: The Face Detection Benchmark" [111], він складається з 32 203 зображення з 393 703 маркованими гранями

Широкий набір даних обличчя - це фактично підмножина більш широкого набору даних [105], який є орієнтиром розпізнавання подій, організованим у 60 класах подій. Щоб побудувати WIDER, категорії подій були вперше визначені та обрані відповідно до великої шкали онтології для мультимедіа (LSCOM) [66]. Потім, використовуючи пошукові системи, такі як Bing або Google, було зібрано від 1000 до 3000 зображень для кожної категорії.

Щоб отримати WIDER FACE з цього набору даних, дані вручну очищали, видаляючи зображення без обличчя людини, а в межах кожної категорії подібні зображення видаляли, щоб забезпечити велику різноманітність зовнішнього вигляду обличчя. Процес анотування проводився шляхом позначення кожного впізнаваного обличчя щільною прямокутною обмежувальною рамкою, що містить лоб, підборіддя та щоки. Для обличь, які дуже важко розпізнати через низьку роздільну здатність або невеликий масштаб (10 пікселів або менше), передбачено прапор "ігнорувати". Більше того, кожне обличчя зазначається своєю позою (типовою чи атиповою) та рівнем оклюзії (жодної, часткової, важкої). Для оклюзії обличчя вважаються частково оклюзійними, якщо 1-30% від загальної площі обличчя оклюзійні та сильно оклюзовані, якщо більше 30%. Пози обличчя позначаються як нетипові за двох умов: або ступінь нахилу більший за 30◦, або відхилення більше 90◦. Є мітки для розмиття, вираження та освітлення. Тим не менш, ці мітки анотуються за допомогою цифр, що супроводжуються одним або двома

словами опису без додаткових деталей. Кожна анотація позначається одним анотатором і двічі перехресно перевіряється різними людьми.

Протокол оцінки точно відповідає протоколу PASCAL VOC [24]. Позитивний вплив цього набору даних полягає в тому, що оцінку можна розділити за різними ознаками. Як згадувалося раніше, набір даних можна спочатку поділити на категорії, що цікаво, оскільки автори класифікували ці категорії за їх складністю. Для кожного з цих класів подій дані поділяються випадковим чином на набори для навчання, валідації та тестування відповідно до таких пропорцій 40%/10%/50%. Обличчя обмежені прямокутними рамками для наборів для навчання та перевірки, але не для тестових наборів. Крім цього, всі обличчя також поділяються на три рівні складності: «легкий», «середній» та «жорсткий» на основі ефективності розпізнавання обличь алгоритму EdgeBox [121]. Базу даних можна підрозділити відповідно до висоти обличчя в пікселях між малими (10-50), середніми (50-300) та великими (> 300).

IARPA Janus Benchmark C (IJB-C) - це нова розширена версія 2017 року бази IARPA Janus Benchmark A (IJB-A), яка була представлена в 2015 році [48] та опублікований Національним інститутом стандартів і технологій США (NIST). Набір даних IJB-C складається з 21 294 зображень із 117 542 обличчями, а також 11 779 відео і спрямований як на розвиток виявлення обличчя, так і на розпізнавання обличчя. IJB-C є третьою версією IARPA Janus Benchmark після IJB-A та IJB-B. На жаль, для IJB-B жодного паперу, подібного до [48] або [101], не було випущено. Оригінальна версія IARPA Janus Benchmark складалася з 5 712 зображень та 2085 відеозаписів з 500 предметів. Цей набір даних був випущений для подолання обмежень багатьох наборів даних про розпізнавання базових обличь, обличчя яких були виявлені за допомогою детекторів обличчя, таких як детектор обличчя Viola & Jones [94]. Для подолання цієї проблеми IJB-A повністю анотувались за допомогою наступної процедури. Була обрана серія з 500 предметів з різним географічним

походженням, а зображення та відео були зібрані шляхом пошуку в Інтернеті за зображеннями ліцензованих Creative Commons. Потім за допомогою щонайменше 5 анотаторів Amazon Mechanical Turk (AMT) в кожному зображенні / кадрі були помічені обмежувальні рамки для обличь. Обмежувальні рамки, що відповідають обличчям були чітко визначені та додатково зазначені місцями очей і носа, кольором шкіри, статі, позою обличчя, оклюзією (очі, рот / ніс, лоб), оточенням (у приміщенні або на вулиці) та обличчям. Загалом це призводить до анотації 67 183 обличчя та до середньої кількості 11,4 зображення та 4,2 відео.

Для оцінки виявлення обличчя зображення в наборі даних розподіляються на 10 навчальних і тестувальних наборів, що вибираються випадковим чином, з двома третинами зображень, доступними для детектора. Точніше, кожен навчальний набір будується шляхом вибірки 333 зображень. Важливо зауважити, що збереглися лише обличчя розміром більше 36 пікселів.

Цей початковий набір даних потім був доповнений для створення другої версії IJB-B. Для отримання цієї нової версії було додано 1345 сюжетів, які призводять до загальної кількості 21 798 зображень (11 754 із обличчями та 10 044 без обличь) та 55 206 кадрів із 7111 відео. Для вибору цих нових предметів 10 000 імен були вперше вилучені з Freebase у 10 різних географічних регіонах. Для кожного з цих кандидатів, кількість доступних Creative Commons (CC) -licensed відео було визначено з допомогою пошуку YouTube API. Потім було завантажено 20 зображень CC для кожного кандидата через зображення Google та Yahoo та розмічено через AMT.

З цієї нової версії прийшли також 10 нових протоколів тестування [101]. Ці протоколи охоплюють оцінки для виявлення особи, перевірки, ідентифікації та кластеризації. Протокол для виявлення обличчя в основному такий же, як для IJB-A. Основна відмінність пов’язана з наявністю тобто зображень, що не містять жодних обличь, в IJB-B. Автори стверджують, що

тестування на зображеннях без обличчя є релевантним, оскільки в реальному застосуванні більшість зображень будуть такого типу.

Таблиці 1.1 та 1.2 узагальнюють інформацію про кожен набір даних через ряд елементів.

Таблиця 1.1 - Опис характеристик орієнтирів виявлення обличь, включаючи загальну кількість зображень у наборі даних, кількість помічених обличь, походження зображень та формат обмежувальної рамки.

Ім'я	Nb зображень	Nb обличчя	Походження	Формат Bbx
FDDB	2845	5,171	Yahoo news website	Еліптичний
AFLW	21,997	25,993	Flickr	Прямокутний / еліптичний
AFW	205	468	Flickr	Прямокутний
Pascal-Faces	851	1335	PASCAL-VOC	Прямокутний
MALF	5250	11,931	Flickr, Baidu Inc.	Квадратний
WIDER FACE	32,303	393,703	Bing, Google search engines	Прямокутний
IJB-C	138,836	> 138 836	CC, Google, Yahoo, YouTube	Прямокутний

Таблиця 1.2 – Опис характеристик орієнтирів виявлення обличчя, включаючи використаний протокол оцінки, рівень точності анотації та тип коду

Ім'я	Протокол	Точність анотацій	Код
FDDB	ROC, specific matching	Низький	Жоден
AFLW	Same as FDDB	Низький	База даних SQlite, графічний інтерфейс міток
AFW	PASCAL VOC	Середній	Навчання та тестування
Pascal-Faces	PASCAL VOC	Низький	Жоден
MALF	PASCAL VOC, ROC	Високий	Жоден
WIDER FACE	PASCAL VOC	Високий	Тестування
IJB-C	ROC	Високий	Невідомо

Розмір бази даних описується з точки зору кількості зображень та обличь для виявлення. Походження зображень та формат обмежувального поля подано більше як спосіб порівняння способу отримання та анотації набору даних. Друга таблиця перераховує протоколи, які використовує кожен тест для оцінювання з рівнем точності, на якій набір даних був анотований (тобто

відносний рівень, на якому кожне поле обмеження було анотовано інформацією, такою як пози, розмиття тощо) та який тип коду надається. На основі цієї інформації ми можемо зробити вибір для орієнтиру оцінки виявлення обличчя.

1.8 Постанововка задачі

За результатами виконаного аналітичного огляду можна зробити висновок, що задача інтелектуального аналізу відеоданих з метою виявлення і ідентифікації особи є достаньо важливою і актуальною, розв’язання якої повинно включати такі етапи:

1) Формування вхідного математичного опису системи інтелектуального аналізу відеоданих
2) Вибір методів детектування обличчя особи
3) Вибір методів ідентифікації особи за виділеним зображенням обличчя
4) Розробка критеріїв оцінки ефективності методів детектування та ідентифікації
5) Розробка та програмна реалізація алгоритмів детектування та ідентифікації
6) Перевірка працездатності системи та порівняння ефективності алгоритмів на задачі детектування та ідентифікації особи

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Методи детектування обличь

FDDB надає список 47 алгоритмів, із найдавнішого - найновішого.

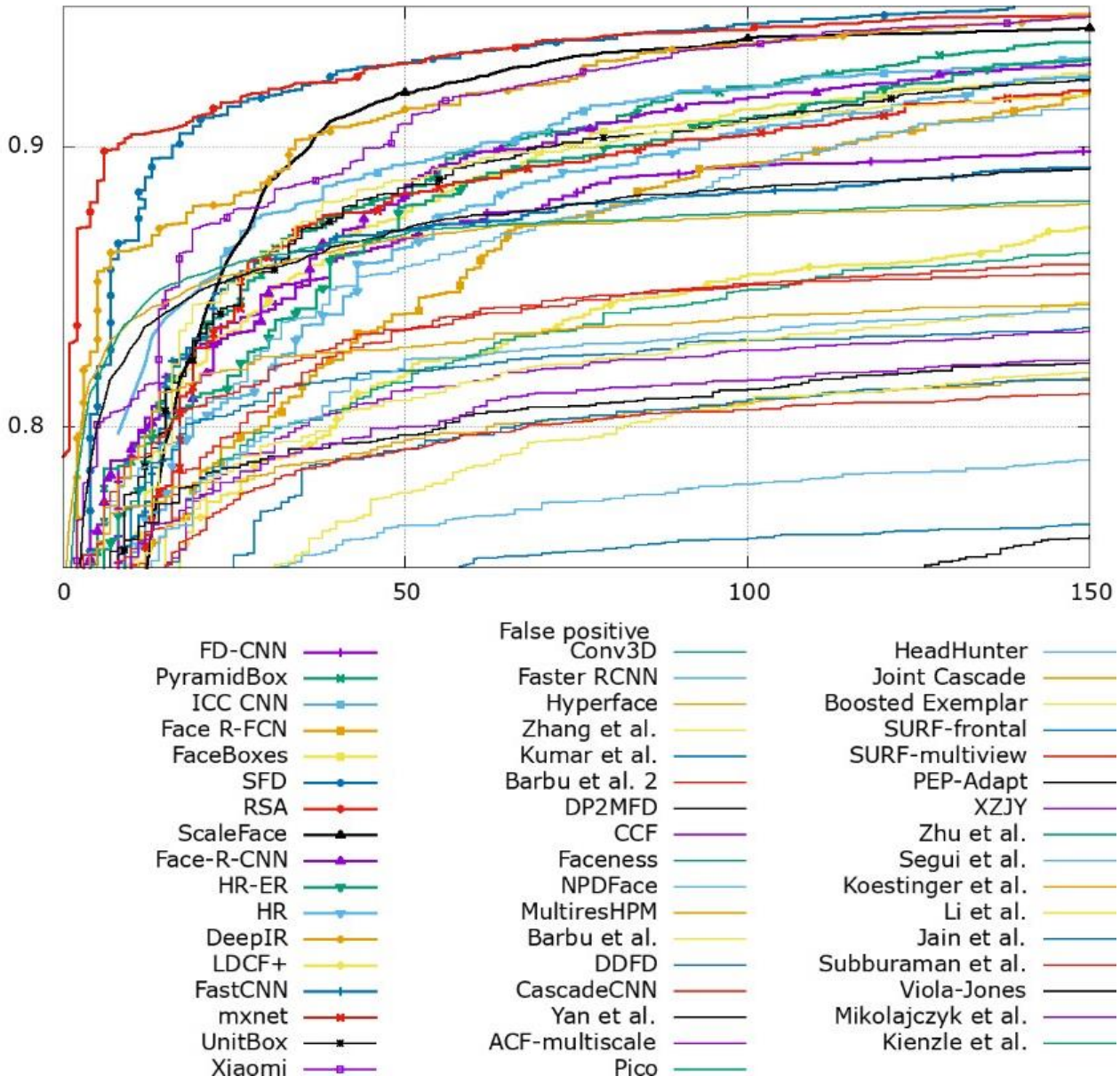


Рисунок 2.1 – ROC криві алгоритмів розпізнавання обличчя на наборі FDDB

Досить складно визначити, наскільки алгоритми працюють відносно один від одного, переглядаючи відображені ROC криві. Дійсно, криві складаються одна над одною, використовуючи кольори для декількох алгоритмів. Таблиця 2.1 містить перелік найкращих методів у приблизному

порядку виконання (починаючи з найкращого методу) із скороченою назвою та дослідницькою роботою, в якій описана методика.

Таблиця 2.1 – Список найкращих опублікованих методів за орієнтиром FDDB

Алгоритм	Опис
RSA	Поточне наближення шкали для виявлення об'єктів у CNN [59]
SFD	SFD: Односхилий масштабний інваріантний детектор обличчя [117]
ScaleFace	Виявлення обличчя за допомогою масштабних глибоких нейронних мереж [112]
DeepIR	Розпізнавання обличь за допомогою глибокого навчання: Покращений швидший підхід RCNN [82]
Xiaomi	Виявлення обличчя при завантаженні за допомогою складних негативних прикладів [95]
ICC-CNN	Виявлення облич за допомогою каскадної контекстуальної нейромережі CNN [115]
Pyramid Box	Pyramid Box: Контекстний детектор обличчя [88]

MALF

У MALF результати представлені також ROC кривими. Для кожних з цих кривих, як показано на рисунку 2.2, площа під кривою також повідомляється у %, що дозволяє легше порівняти ефективність алгоритму.

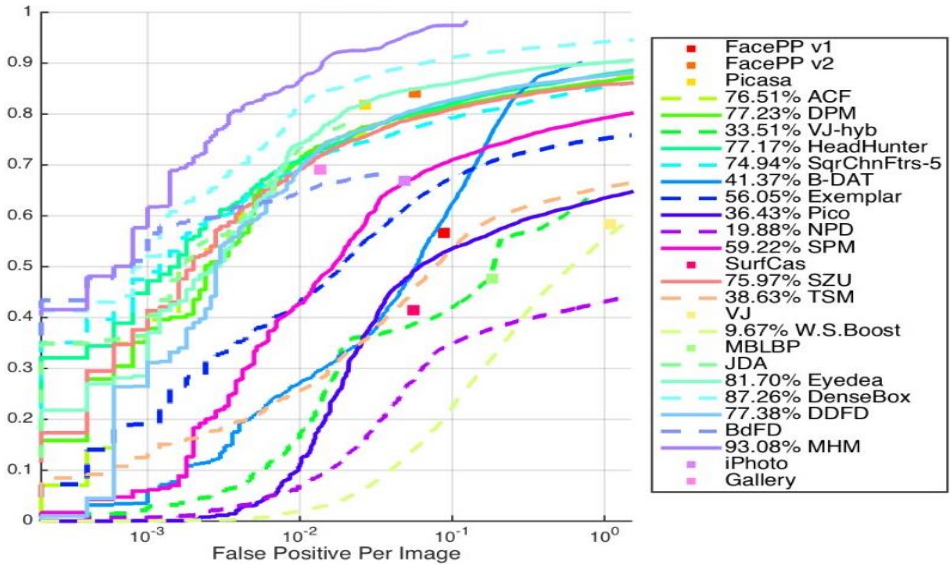


Рисунок 2.2: - ROC-криві, отримані алгоритмами розпізнавання обличчя на цілому еталоні MALF

Таблиця 2.2: - Список найкращих опублікованих виконавців за еталоном MALF

Алгоритм	Опис
JDA	Спільне виявлення та вирівнювання обличчя каскаду [16]
DDFD	Багатовимірне розпізнавання обличь за допомогою глибоких нейронних мереж [25]
DPM	Виявлення обличчя без конусів [63]
HeadHunter	Виявлення обличчя без конусів [63]
ACF	Функції сукупного каналу для розпізнавання обличь у кількох видах [107]
SqrChnFtrs-5	Виявлення обличчя без конусів [63]

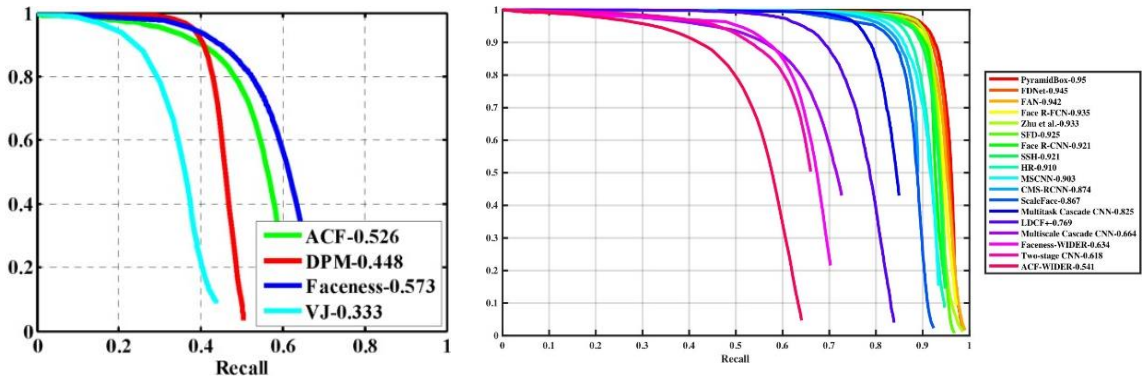
Якщо ми проаналізуємо різні графіки, ми можемо легко побачити, що серед топ - 12 кращих алгоритмів 6 аналогічні попередній таблиці. Таблиця 2.2 показує цей список в порядку виконання (починаючи з найкращого алгоритмому) на основі тесту всього набору даних з їх скороченим ім'ям і в відповідної статті.

WIDER FACE

Щодо WIDER FACE, існує також два списки алгоритмів. Перший список тестується в Scenario-Ext (тобто детектор обличчя навчається за допомогою будь-яких зовнішніх даних, що тестується на наборі WIDER FACE) для легких, середніх і жорстких підмножин. Результати відображаються з використанням ROC кривих з додатково середньою точністю для кожного алгоритму. Цей список містить тільки 4 алгоритми рис. 2.3 а):

Таблиця 2.3 – Список алгоритмів розпізнавання обличь, протестованих на еталоні WIDER FACE з протоколом Scenario-Ext.

Алгоритм	Опис
Faceness	Від відповідей частин обличчя до розпізнавання обличь: глибокий підхід до навчання [110]
ACF	Функції сукупного каналу для розпізнавання обличь у кількох видах [107]
DPM	Виявлення обличь без конусів[63]
VJ	Впровадження OpenCV детектора обличчя Viola-Jones [94]



а б

Рисунок 2.3 – ROC – криві: а) Scenario-Ext і б) Scenario-Int на 07-06-2018.

Маючи результати для трьох підмножин ми можемо легко визначити 10 найкращих методів, які досягають подібних показників (AP> 90% на легких та середніх наборах та AP> 80% на жорсткому наборі) як на валідаційному наборі, так і на тестовому наборі. Перелік цих алгоритмів, класифікованих по продуктивності в середньому наборі, показано на рисунку 2.4, як показано в таблиці 2.4.

Таблиця 2.4 - Список найкращих методів за тестовим показником WIDER FACE Scenario-Int

Алгоритм	Опис
PyramidBox	PyramidBox: Контекстний детектор обличчя з одною стрілкою [88]
FDNet	Не посилається
FAN	Мережа пошуку обличчя: Ефективний детектор обличчя для закритих облич [98]
Zhu et al.	Small Faces from Robust Anchor’s Perspective [119]
Face R-FCN	Виявлення обличчя за допомогою конволюційних мереж на основі регіону [99]
SFD	SFD: Односхилий масштабний інваріантний детектор обличчя [117]
Face R-CNN	Face R-CNN [98]
SSH	SSH: Одностадійний детектор обличчя [65]
HR	Пошук малих облич [40]
MSCNN	Єдина багатомасштабна глибока згорткова нейронна мережа для швидкого виявлення об'єктів [11]

В таблиці 2.7 наведено перелік усіх алгоритмів, які вважаються вартими перевірити. У цей список додано інформацію про такий критерій відбору алгоритму як відкритий код. Щоб заощадити простір, URL-адреси сховищ явно не пишуться, але доступні через гіперпосилання під ключовими словами "Офіційні / Неофіційні".

Таблиці 2.5 - Список алгоритмів виявлення обличчя для тестування.

Алгоритм	Код
ACF	Не офіційний
DDFD	Не офіційний
DPM	Офіційний
Face R-CNN	Не офіційний
Face R-FCN	Не офіційний
FAN	-
HeadHunter	Офіційний
HR	Офіційний
ICC-CNN	-
JDA	-

Алгоритм	Код
MSCNN	Офіційний
PyramidBox	-
RSA	Офіційний
ScaleFace	-
SFD	Офіційний
SqrChnFtrs-5	-
SSH	Офіційний
VJ	Офіційний
Xiaomi	-
Zhu et al.	-

З таблиці 2.5 не видалявся алгоритм VJ - детектор OpenCV Haar-Cascade, що походить від алгоритму Viola&Jones [94], оскільки він вважається першим практичним детектором обличчя і досі широко використовується. На додаток до цих алгоритмів, було вирішено також протестувати детектор Dlib на основі методу HOG [19].

2.2 Методи ідентифікації особи

LFW

В таблиці 2.4 подано 10 основних методів ідентифікації особи на бази тестових даних LFW за протоколом "Unrestricted Labeled Outside Data".

Таблиця 2.4 - Список найкращих, опублікованих виконавців на наборі LFW з протоколом "Unrestricted Labeled Outside Data"

Алгоритм	Опис
Tom-vs-Pete*	Класифікатори Tom-vs-Pete та збереження ідентичності для перевірки обличчя [9]
High-dim LBP	Ефективне стиснення для перевірки обличчя [15]
TL JB*	Практичний алгоритм перевірки обличчя [13]
DeepFace*	DeepFace: закриття прогалини в продуктивності на рівні людини при перевірці обличчя [87]
POOF*	Особливості "один на один" для категоризації, перевірки обличчя та оцінки атрибутів [8]
DeepID	Поглиблене представлення особи із передбачуванням 10 000 класів [83]
FaceNet	FaceNet для розпізнавання та кластеризації обличчя [74]
MMDFR	Розпізнавання обличчя за допомогою Multimodal Deep Face Representation [23]
P+S+E Aug.*	Чи дійсно нам потрібно зібрати мільйони обличь для ефективного розпізнавання обличчя? [62]
Dlib-R*[3]	Реалізація впровадження глибокого залишкового навчання для розпізнавання зображень [20]

MegaFace

У MegaFace важливі результати, отримані у наборі даних FaceScrub.

Таблиця 2.5 - Список найкращих виконавців, визначених на MegaFace Challenge 1, протокол ідентифікації з FaceScrub.

Алгоритм	Опис
ArcFace	ArcFace: додаткова кутова грань для глибокого розпізнавання обличчя [21]
SphereFace	SphereFace: глибока вбудованість гіперсфери для розпізнавання обличчя [57]
FaceNet	FaceNet: Єдине вбудовування для розпізнавання та кластеризації обличчя [74]

У таблиці 2.6 подано алгоритми двох попередніх списків і зазначено, який із них не забезпечує реалізацію з відкритим кодом, а якщо він є, то ця реалізація є офіційною чи неофіційною.

Таблиця 2.6: - Список потенційних алгоритмів розпізнавання обличчя для тестування.

Алгоритм	Код
ArcFace	Офіційний
DeepFace	-
DeepID	Неофіційний
Dlib-R	Неофіційний
FaceNet	Неофіційний
High-dim LBP	Неофіційний

Алгоритм	Код
MMDFR	-
P+S+E Aug.	Неофіційний
POOF	-
SphereFace	Неофіційний
TL JB	-
Tom-vs-Pete	-

2.3 Опис алгоритмів детектування

У цьому підрозділі описано алгоритми, які будуть перевірені далі. Крім того, оскільки деякі розробки алгоритмів іноді пов'язані один з одним, опис сортується в хронологічному порядку публікації методів.

Моєю початковою метою було перевірити всі алгоритми, для яких я знайшов офіційну чи неофіційну реалізацію. Однак через брак часу я не досягнув цієї мети. Тому я спершу зосередився на алгоритмах з офіційними реалізаціями. На жаль, мені не вдалося запустити їх усіх. Проблеми, з якими я зіткнувся прийшли головним чином з міркувань сумісності з CUDA і cuDNN або у випадку алгоритмів, написаних у Matlab, систематичні фатальні помилки при запуску алгоритмів. Всього мені вдалося запустити 5 алгоритмів виявлення обличчя та 3 алгоритми розпізнавання обличь.

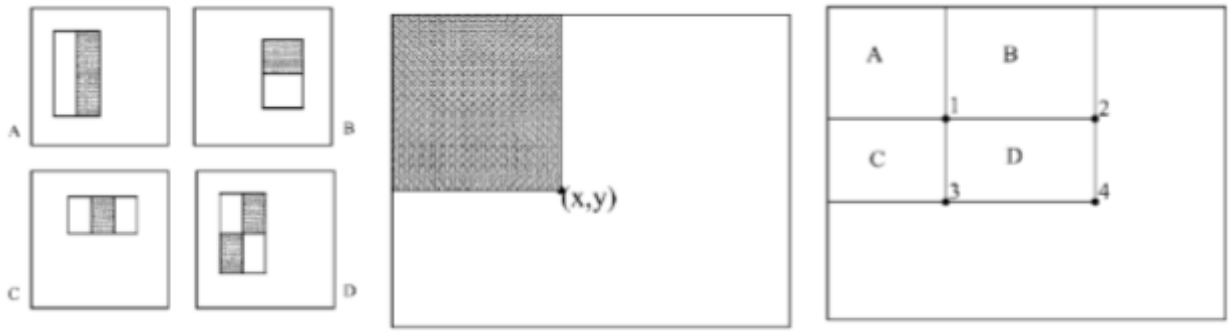
П'ять алгоритмів, які будуть описані для виявлення обличчя, - це VJ, HOG, FRCNN, SFD та SSH.

2.3.1 VJ

Алгоритм, запропонований Віолою та Джонсом [93] та вдосконалений у [94], був одним із перших методів виявлення об'єкта (та обличчя) у реальному часі. Алгоритм ґрунтується на трьох основних поняттях: характеристики та інтегральне зображення, вибір функції за допомогою Adaboost [27] та каскаду. В результаті, ковзне вікно виділяє частину зображення, на якому виконується розпізнавання обличчя, і для кожного положення вікна обчислюється значення

певної функції. Потім ці функції передаються через каскад класифікаторів, щоб визначити, чи відповідають ці пікселі обличчю.

Функції, які використовуються в алгоритмі Viola&Jones обчислюються за різницею в сумі пікселів різних частин (2, 3 і 4) суміжних прямокутників. Для того, щоб обчислити функцію більш ефективно, на даній шкалі сірого зображення, вона спочатку перетворюються в «інтегральний образ», де пікселі в положенні (x, y) є просто сумою значень пікселів вгорі та зліва від (x, y) у вихідному зображенні



а) б) в)

Рисунок 2.4 – Метод Віолли-Джонса [94] а) приклад функцій прямокутника, показаних відносно вікна. Сума пікселів, що лежать всередині білих прямокутників, віднімається від суми пікселів у сірих прямокутниках. Особливості двокутника показані в (A) та (B). На рисунку (C) показана трикутна особливість, а (D) – чотирикутна;

б)Значення інтегрального зображення в точці (x, y) - це сума всіх пікселів вгорі та зліва; в) Суму пікселів у прямокутнику D можна обчислити чотирма посиланнями масиву. Значення інтегрального зображення в розташуванні 1 - це сума пікселів у прямокутнику A. Значення в розташуванні 2 - A + B, у місці 3 - A + C, а в місці 4 - A + B + C + D. Суму в межах D можна обчислити як 4 + 1 - (2 + 3)

Обчислені значення використовує класифікатор, щоб розрізняти зображення обличчя і зображення без обличчя. Вибраний у [94] класифікатор - це модифікована версія Adaboost. Для підвищення продуктивності алгоритм був модифікований заміною простого класифікатора каскадом класифікаторів. Цей каскадний класифікатор працює аналогічно «дереву рішень». За допомогою цієї методики кінцева модель використовує 6060 функцій, розділених на 38 шарів каскаду класифікаторів, зберігаючи високу частоту кадрів. На рисунку 2.5 показаний приклад двох перших обраних функцій.

В [94] висуваються два параметри. Вони відносяться до процесу ковзного вікна, в якому детектор сканується над зображенням. Дійсно, обличчя можуть бути розміщені в будь-якому місці зображення та в різних масштабах.

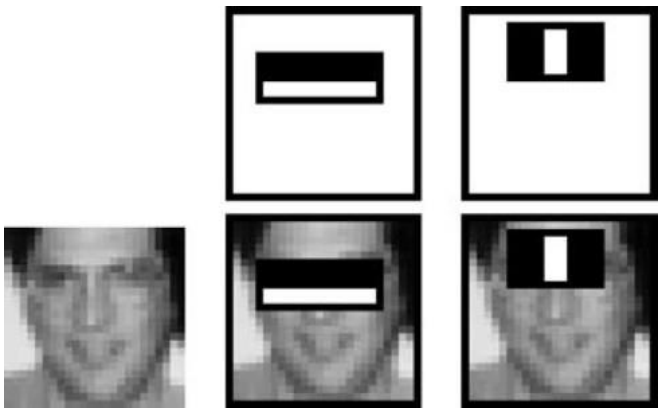


Рисунок 2.5 - Метод Віолли-Джонса [94]

Перша та друга функції, обрані AdaBoost. Дві функції відображаються у верхньому ряду, а потім накладаються на одне з зображень обличчя з навчальноївибірки. Перша - вимірює різницю в інтенсивності між областю очей і областю щік. Ця особливість використовує той факт, що область очей часто темніша за щоки. Друга особливість порівнює яскравість пікселів в області очей і носа.

Процес масштабування управляється шляхом масштабування розміру детектора (чий мінімальний розмір 24x24 в оригінальній роботі). Для переміщення детектора по зображенню можна просто використовувати крок в

один піксель або збільшити цей крок. Чим менший цей крок, тим більше осіб може бути виявлено, але повільніше алгоритм.

Крім того, у статті розглядається проблема множинних виявлень одного обличчя. Оскільки кінцевий детектор є нечутливим до невеликих змін в позиції і масштабі, кілька виявлень, як правило, відбуваються навколо кожного обличчя в від сканованому зображенні. Вирішення цієї проблеми полягає в групуванні всіх виявлень. Причому два виявлення знаходяться в одній підмножині, якщо їхні обмежуючі області перетинаються. Тоді для кожної з цих підмножин виробляється одна обмежуюча область, кути якої є середнім кутом усіх виявлень у наборі.

2.3.2 HOG

Ідея цього HOG алгоритму полягає в тому, що зовнішній вигляд та форму локальних об'єктів часто можна досить добре характеризувати розподілом локальних градієнтів, навіть без точного знання відповідних позицій градієнта. Завдання алгоритму полягає в тому, щоб створити вектор функцій для даної системи зображення на основі цих градієнтів і використовувати ці функції для класифікації систем. Повна процедура вилучення можливостей може бути просто перенесена на виявлення обличчя, лише різниця полягає в тому, що новий класифікатор повинен бути навчений за відпвідними начальними вибірками.

Перший крок алгоритму полягає в розділенні зображення на ряд малих просторових областей, які називаються осередками, розміром 8x8 пікселів. Для кожної з цих осередків обчислюється 1-D гістограма градієнтних напрямків. Ідея полягає у застосуванні простої [1, 0, 1] маски над пікселями комірки, щоб отримати інтенсивність градієнта в кожному напрямку. Потім кожен піксель обчислює зважене значення для каналу гістограми, орієнтуючись відповідний градієнтний елемент, і значення, накопичені у осередках. Алгоритм міг би зупинитися на цьому, використовуючи конкатенацію гістограм усіх комірок як векторної ознаки, але автори

застосовують спочатку деяку нормалізацію до даних. Техніка їх нормалізації заснована на групуванні осередків у більші просторові блоки та контрасти, що нормалізують кожен блок окремо. У конфігурації за замовчуванням кожен цей блок складається з 16x16 пікселів, що містять 4 комірки. Крім того, блоки, як правило, перекриваються, тобто кожне скалярне значення осередка вносить декілька компонентів у кінцевий вектор дескриптора, кожен з яких нормалізується щодо іншого осередка. У конфігурації за замовчуванням кроковий проміжок між осередками має 8 пікселів, тобто кожна комірка перекрита 4 рази. Під час фази тестування вікно виявлення ковзає по зображенню і застосовує цей алгоритм у кожній позиції.

2.3.3 FRCNN

Face R-CNN [97] підхід заснований на методі виявлення об'єктів "Faster R-CNN " [71]. Faster R-CNN (FRCNN) - це основне оновлення методу R-CNN [29]. Впровадження цього методу стимулювало розробку низки підходів, заснованих на CNN. R-CNN розшифровується як області з функціями CNN. Коли цей метод був запропонований, найсучасніші методи виявлення об'єктів базувалися на особливостях HOG або SIFT [61]. Як представлено в попередньому розділі, ці методи використовують підхід ковзного вікна, в якому для кожного положення вікна генерується вектор функцій, а потім класифікується за допомогою якогось класифікатора (наприклад, SVM). У цьому випадку CNN також застосовувались як екстрактор функцій. У [97] пропонується альтернатива підходу з ковзним вікном для проблеми локалізації, яка називається парадигмою «розпізнавання за допомогою області»[32]. Основна ідея полягає в заміні кроку ковзання вікна на крок пропозиції регіону, який генерує набір незалежних від категорії пропозицій області для вхідного зображення.R-CNN структура буде складатися з трьох модулів, як проілюстровано на рисунку 2.13: Перша категорія формує незалежну область пропозиції. Ці пропозиції визначають набір виявлення кандидатів, доступних нашому детектору. Другий модуль - це велика

нейронна мережа, яка витягує функціональний вектор фіксованої довжини з кожної області. Третій модуль - це набір лінійних SVM, специфічних для класу.

Рисунок 2.6 - Алгоритм R-CNN [29]

Незручною вимогою R-CNN є те, що вхідне зображення CNN повинно бути фіксованого розміру [35]. Рішенням, запропонованим [35], є введення в CNN шару "просторового об'єднання пірамід" (SPP), як показано на рис.2.14. Методика полягає в тому, щоб додати шар SPP поверх останнього згорткового шару. Шар SPP об'єднує функції в локальних просторових елементах, що мають розміри, пропорційні розміру зображення, і генерують виводи фіксованої довжини, які потім подаються у повністю зв’язані шари (або інші класифікатори). Автори називають свою мережу за допомогою цієї методики SPP-Net. Ще одна проблема R-CNN полягає в тому, що обчислення займають багато часу, оскільки в них неодноразово застосовують глибокі згорткові мережі. SPP-net значно покращує цей обчислювальний час (від10 до 100 разів Faster за тестовий час) за рахунок обчислення згорткової карти характеристик для всього вхідного зображення, а потім класифікації кожної пропозицію об'єкта, використовуючи векторний елемент, що сформовано із спільної карти функцій.

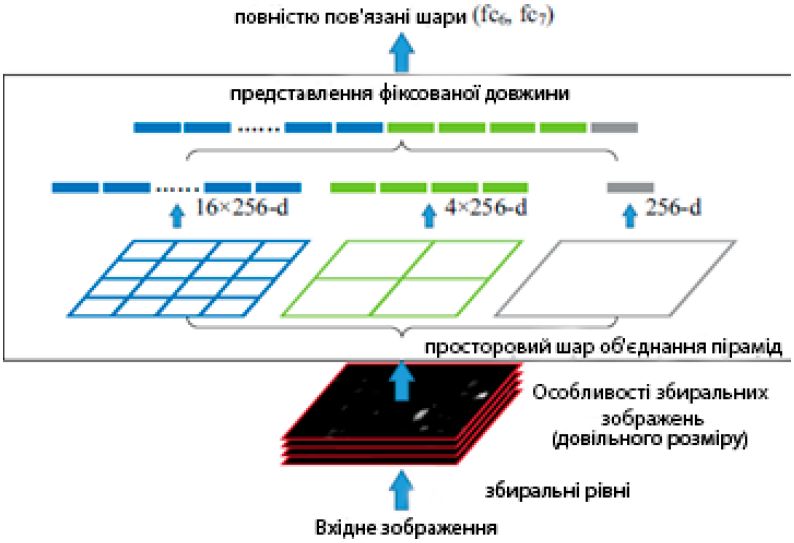


Рисунок 2.7 - Введення SPP-шару в CNN[35]

Основна ідея FRCNN полягає в тому, щоб об’єднати CNN з двома вихідними шарами двійників, які сформують максимальні оцінки ймовірності для класів появи K об'єктів, загальнодоступний клас "фон" та інший рівень, який виводить чотири дійсні значення числа для кожного з класів об'єктів K. Кожен набір із 4 значень кодує положення обмежувальної рамки для одного з K класів. Перший шар знімає потребу в класифікаторі SVM, тоді як другий шар виконує регресію обмежувальної рамки. Як доповнення, в [28] замінено рівень SPP на "ROI-шар", який є особливим випадком шару SPP, в якому є лише один рівень піраміди. Використовуючи цю техніку, швидкий R-CNN може тренувати глибокі мережі в 9 разів швидше, ніж R-CNN і в 3 рази швидше, ніж SPP-Net.

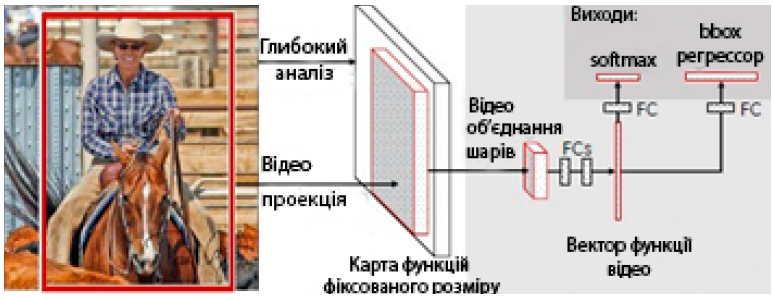


Рисунок 2.8 - Швидка конструкція R-CNN[28]

2.3.4 SFD

Індикатор обличчя індексованого масштабу (SFD) представлений в роботі [117] у вигляді реального детектора обличчя в режимі реального часу, який здатен детектувати обличчя різних масштабах за допомогою єдиної глибокої нейронної мережі. З одного боку, ця мережа дотримується нового підходу, запропонованого в Faster R-CNN щодо використання так званих анкерів, з яких визначено пропозиції області. Крім того, автори згадують, що вони також були натхнені іншою гілкою моделей на основі анкерів, прикладом яких є метод Single Shot MultiBox Detector (SSD), введений в [56]. З іншого боку, автори SFD критикують ті методи, особливо на предмет виявлення частин обличчя.

SFD базується головним чином на Faster R-CNN в тому, що він використовує «згорткові пріоритети (або анкерні рамки)». У випадку Faster R-CNN ці анкери використовуються для виготовлення "рамки-регресії" (тобто визначення пропозицій області, параметризованих відносно анкерів) та "класифікації рамки" (тобто класифікує відповідну пропозицію як об'єкт чи не об'єкт) за допомогою мережі RPN. Об'єкт частина виявлення мережі потім використовує ці пропозиції, щоб передбачити, чи містять вони об'єкт інтересу. Виходячи з мереж YOLO [70] та OverFeat [77], ідея SFD полягає в об'єднанні цих двох кроків в один, шляхом безпосереднього. Таким чином, такий підхід дозволяє уникнути ускладнення злиття RPN з Fast R-CNN . Тому основна ідея полягає не просто в тому, щоб передбачити, чи є регіональна пропозиція об'єктом чи не об'єктом, а безпосередньо дати їй оцінку для кожного розглянутого класу. Як показано на рисунку 2.9, SFD визначає три причини, спільні для обох Fast R-CNN та SSD, які роблять їх неефективними у виявленні дрібних предметів та пропонують рішення цих проблем. По-перше, автори вважають, що існує "упереджена структура". Дві основні ідеї полягають у тому, що розмір кроку найнижчого пов’язаного з анкером шару (тобто карта характеристик, пов’язана з анкорем та з якої області будуть генеруватися

пропозиції), занадто велика і що масштаб анкора занадто великий для маленьких обличь. Для авторів це негативно вплинуло на виявлення малих та середніх обличь. Їх рішення полягає в тому, щоб запропонувати справедливу за масштабом рамку виявлення обличчя. Це рішення складається в основному з анкорів, що використовують більші розміри кроки, і для розробки шкали анкорів відповідно до сприйнятливого поля.

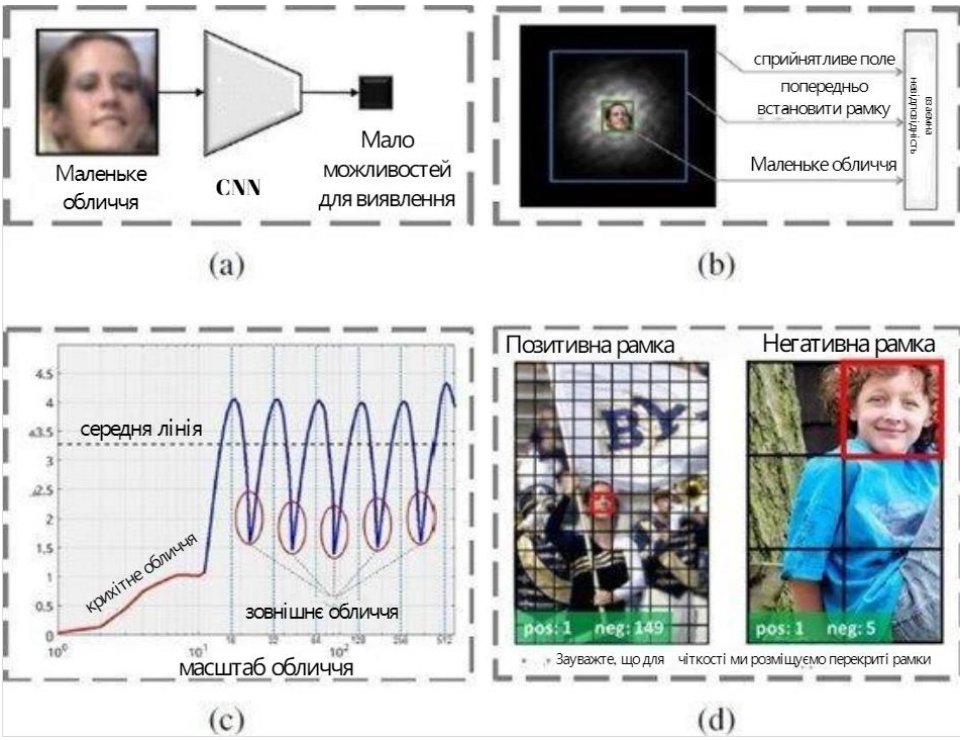


Рисунок 2.9 - Недоліки анкорних методів: a) Мало можливостей: Маленькі обличчя мають мало можливостей на рівні виявлення. b) Невідповідність: розміру анкорів сприйнятливому полю, і обидва є занадто великими, щоб підходити до маленького обличчя.c) Стратегія узгодження анкора

На рисунку показано кількість відповідних якорів на різних масштабах обличчя за поточним методом узгодження якорів. Це відображає, що крихітні та зовнішні грані відповідають занадто мало якорів. d) Передумови маленьких якорів: дві фігури мають однакову роздільну здатність. Ліва плитка - маленькі якорі для виявлення малого обличчя, а права - великі якорі для виявлення

великого обличчя. Невеликі якорі створюють безліч негативних якорів на задньому плані

Другий недолік, на який вказується, - це невідповідність дискретних анкорів шкал та суцільної шкали обличчя, проілюстрована на рисунку в [117]. Щоб покращити це, автори пропонують стратегію узгодження компенсації анкорів з двома етапами.

Перший етап дотримується поточного методу узгодження анкера, але коригує більш розумний поріг. Другий етап гарантує, що кожна шкала граней відповідає достатній кількості анкорів за рахунок компенсації шкали. Нарешті, спроба виявити невеликі обличчя створює нову проблему, яка полягає в тому, що велика кількість маленьких анкорів повинна щільно покривати зображення, що призводить до різкого збільшення кількості негативних анкорів на фоні, що призводить до появи багатьох помилкових позитивних рішень. Рішенням цього є використання максимального рівня фону для найнижчого рівня виявлення.

2.3.5SSH

“SSH: Single Stage Headless Face Detector” [65] пропонує аналогічний підхід до SFD в тому сенсі, що це також одноетапний детектор, який безпосередньо формує висновок для кожного класу по кожному з пропозицій області.

Дві додаткові особливості SSH полягають у тому, що він є інваріантним за масштабами дизайну і може ефективно включати контекст. Щоб виділяти обличчя різних масштабів, автори беруть інформацію [55] шляхом застосування модулів виявлення на трьох різних згорткових шарах, як показано на рисунку 2.10. Вихід цих трьох модулів аналогічний виходу SSD з координатами рамок і балами для кількох категорій. Ми можемо бачити, що цей підхід сильно відрізняється від SFD, де методика полягає в основному у множенні анкорів за рахунок зміни їх розмірів та кроків.

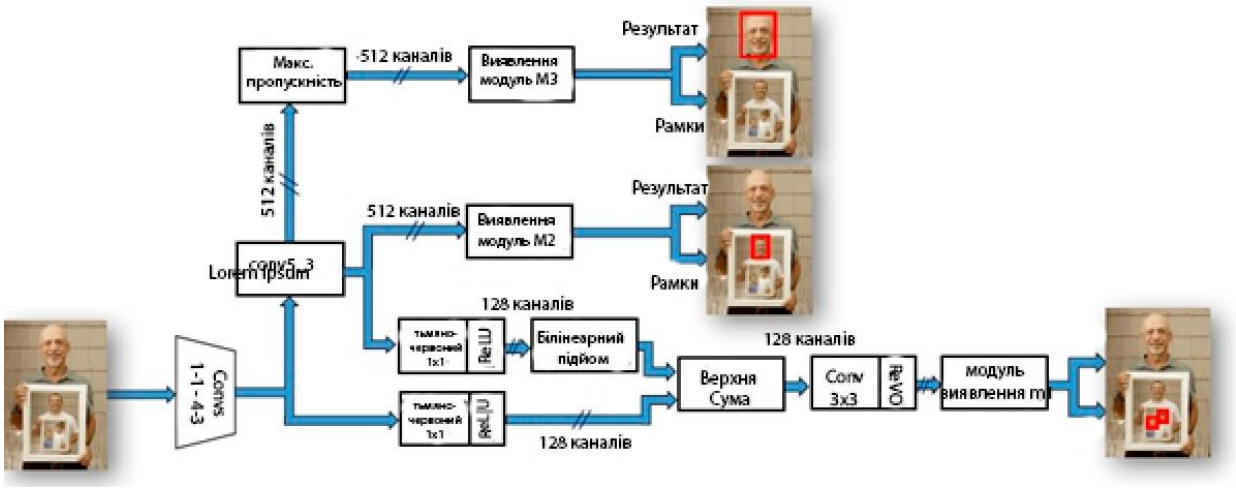
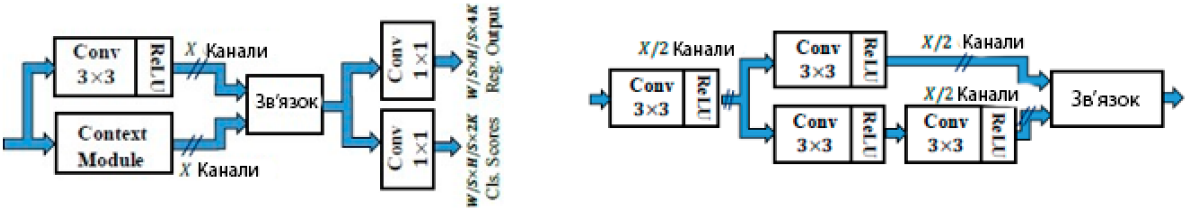


Рисунок 2.10 – Мережева конструкція SSH [65]

SSH реалізує контекстуалізацію безпосередньо в мережі шляхом інтеграції згорткового контекстного рівня в модулі виявлення, як показано на рисунку 2.11 а). Цей контекстний модуль полягає головним чином у застосуванні більшого фільтра розміром 5x5 та 7x7, а не 3x3. Однак, як показано на рисунку 2.11 б), ці великі згорткові фільтри замінюються послідовними фільтрами 3x3 з метою зменшення кількості параметрів, що є методом, подібним до того, який запропоновано для мережі Inception в [86].



а б

Рисунок 2.11 – Модулі виявлення[65] (а) та контексту (б).

Основна реалізація SSH використовує як вхідні зображення, зменшуючи розмір найкоротшої сторони до 1200 пікселів, зберігаючи найбільшу сторону нижче 1600 пікселів, не змінюючи співвідношення сторін. Інша можливість - використовувати SSH з пірамідою зображення. Ідея полягає в тому, щоб

просто застосувати виявлення до вхідного зображення, розміщеного за розміром, зі змінною кількістю різних розмірів.

2.4 Опис алгоритмів ідентифікації

Три алгоритми, описані в цьому підрозділі, називаються OpenFace, Dlib-R і ArcFace. Ці три алгоритми ґрунтуються на конструкціях глибоких нейронних мереж, їх мета - проектувати зображення у просторі роздільних функцій, де схожі об’єкти кластеруються та відокремлюються від різних об’єктів.

2.4.1OpenFace

Як зазначено в роботі [5], бібліотека розпізнавання обличь OpenFace реалізує два підходи глибокого навчання під назвою DeepFace [87] та FaceNet [74]. FaceNet - це система, яка безпосередньо виконує відображення зображень обличчя в компактний євклідовий простір, де відстані безпосередньо відповідають мірі подібності обличчя. Екстракція ознак здійснюється за допомогою глибокої нейронної мережі, як у попередніх роботах, таких як DeepFace [87] або DeepId2 + [103]. Автори уточнюють, що їх метод досягає більшої репрезентативної ефективності, оскільки використовує так звані триплетні втрати. Ідея триплетної втрати полягає в тому, щоб «зображення x^a_i (анкір) конкретної людини було ближче до всіх інших зображень x^p_i (позитивних) тієї самої людини, ніж до будь-якого зображення x^n_i (негативного) будь-якої іншої людини». Математично це означає, що для кожної триплети ми хочемо, щоб відстань між вкладенням x^a_i та x^p_i була евклідовою, щоб вона була строго меншою в межах постійної, ніж евклідова відстань між x^a_i та x^n_i.

$$\left\|x_i^a - x_i^p\right\|_2^2 + a < \left\|x_i^a - x_i^n\right\|_2^2 \forall \left(x_i^a, x_i^p, x_i^n\right) \in T \qquad (2.1)$$

Набір усіх можливих трійників міститься у навчальному наборі

Далі процес навчання застосовується до заздалегідь визначеної конструкції нейронної мережі. Зокрема, тестуються дві наступні конструкції. Перша заснована на моделі Zeiler&Fergus [114]. Ця модель складається з декількох перемежованих згортків шарів, нелінійних активацій, локальних нормалізацій відповіді та максимуму об'єднаних шарів і доповнюється додаванням декількох шарів згортання між стандартними згортковими шарами. Другий базується на моделях «GoogleNet style Inception [86]», в яких використовуються змішані шари, які паралельно керують декількома різними згортковими та об’єднаними шарами.

Нарешті, майже жодна попередня обробка не застосовується до зображень перед тим, як їх подавати в мережу під час навчання чи тестування.

2.4.2Dlib-R

Dlib - бібліотека з відкритим кодом, що пропонує ряд алгоритмів машинного навчання. Як зазначалося на його веб-сайті, “Dlib - це сучасний інструментарій C++, що містить алгоритми машинного застосування та інструменти для створення складного програмного забезпечення на C++ для вирішення реальних проблем. Він використовується як в промисловості, так і в наукових колах в широкому діапазоні областей, включаючи робототехніку, вбудовані пристрої, мобільні телефони та великі високоефективні обчислювальні середовища. Під час оновлення, намагаючись запровадити глибокий метричний інструмент навчання, до бібліотеки було додано програму розпізнавання обличь.

Як і у OpenFace, мета системи розпізнавання обличчя полягає у створенні зображень обличчя з низькими розмірами. В основі нейромережевої конструкції лежить мережа ResNet з 29 згортковими шарами. В основному вона складається в спрощеній версії ResNet-34 [36].

2.4.3ArcFace

ArcFace - це назва функції втрат та алгоритму, який використовує її [21]. Для цього вони визначають три основні атрибути, що розрізняють моделі розпізнавання обличь: навчальні дані, використовувані для формування моделі, мережеву конструкцію, налаштування та вид функції втрат.

Що стосується даних про навчання, то основна робота полягала у вдосконаленні одного з обраних даних навчального набору, MS-Celeb-1M. Цей процес очищення був зроблений напівавтоматичним. Основна ідея полягала в тому, щоб використовувати алгоритм розпізнавання обличчя для ранжування всіх зображень обличчя кожної особи за їх відстані до центру ідентичності та автоматичного видалення зображень обличчя, чий функціональний вектор занадто далеко від центру функцій особи [22]. Більше того, зображення обличчя, що були біля порогу, перевіряються вручну. Цей вдосконалений набір даних використовується для підготовки остаточної моделі, яка тестується на наборі даних MegaFace. Однак набір даних VGG2 також використовується для навчання інших мереж для проведення низки експериментів. Тести на MegaFace фактично проводяться на двох версіях цього еталону: оригінальній та вдосконаленій. Автори показують на прикладах, що в FaceScrub є шум (тобто обличчя, пов’язані з неправильною ідентичністю), і що між FaceScrub та набором даних MegaFace над ідентичністю існує перекриття.

Потім автори порівнювали різні мережеві конструкції, змінюючи ряд параметрів. Усі ці мережі дотримуються деякої загальної структури. Усі вони приймають за вхід RGB-зображення розміром 112x112, отримані вирівнюванням, обрізанням, зміною розміру та нормалізацією вихідних зображень обличчя. Перша різниця пов'язана з розміром виходу, який можна встановити на 7x7 (позначається L на початку мережі) або 3x3. Потім перевіряється 5 різних варіантів на встановлення вбудовування. Потім тестуються дві різні версії мережі ResNet. Ряд інших мережевих архітектур

порівнюється, включаючи MobileNet [38], Inception-ResNet-V2 [85], DenseNet [41], мережі стискання та збудження [39] та мережу подвійного шляху [17]. Кінцевою моделлю, яку вони використовують для порівняння продуктивності між різними втратами, є LResNet100E-IR.

Підсумковий тест полягає у порівнянні продуктивності алгоритму з використанням різних функцій втрат. Перед цим автори роблять огляд різних функцій втрат, розроблених для розпізнавання обличь за допомогою глибокого навчання. Ці втрати можна розділити на два основні класи. Більш ранній клас, який автори називають «втрати на основі евклідової маржі», містить найменші втрати, що використовуються, наприклад, у DeepFace. Альтернативою тим втратам у тому ж класі є приклад триплетної втрати [74] або контрастної втрати [84], яка використовує стратегію парних тренувань, при цьому питання полягає у виборі ефективних навчальних зразків. Другий клас втрат ґрунтується на модифікаціях втрат програмного забезпечення [58]. [96] є зразками цього класу, і ArcFace також є його частиною.

3 ІНФОРМАЦЙНЕ І ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Цей розділ містить низку результатів, отриманих за допомогою оцінки алгоритмів виявлення обличчя та розпізнавання обличчя, які були описані в попередньому розділі за двома наборами тестових даних, які були обрані розділі 1.7.

3.1 Детектування обличь

3.1.1 Протокол оцінювання

На основі елементів, проаналізованих у підрозділі 1.7, було вирішено використати WIDER FACE як набір для навчання і тестування. WIDER FACE постачається з низкою Matlab функцій для оцінки та побудови графіків. На додаток до цього, три файли '.mat' містять структури даних з зображеннями облич, що розбиті на підмножини 'easy', 'medium' та 'hard' та метадані, пов'язані з кожним обличчям. Загальна кількість обличь для виявлення в кожній підмножині становить 7,211, 13,319 та 31,958 для легкого, середнього та жорсткого підмножини відповідно.

Кожне обличчя пов’язане з одним значенням для кожного з наступних атрибутів: розмиття (ясне: 0, нормальне розмиття: 1, сильне розмиття: 2), вираз (типово: 0, перебільшення: 1), освітленість (нормальне: 0, крайнє: 1), оклюзія (немає: 0, часткова: 1 або важка: 2) і нахил (типова: 0 або нетипова: 1).

На рис. 3.1 показано, скільки обличь зазначається з кожним значенням (від 0 до 1 або 0 до 2) з 6 різних функцій. Для кожного з цих значень кольорові смуги вказують, скільки граней, пов’язаних з цим значенням , міститься у кожній з трьох підмножин WIDER FACE.

Нарешті, прапор (дійсний: 0, недійсний: 1) пов’язаний з кожним обличчям. Однак жодне з обличь у підмножинах не має значення 1, тому нам не потрібно звертати увагу на цей прапор.

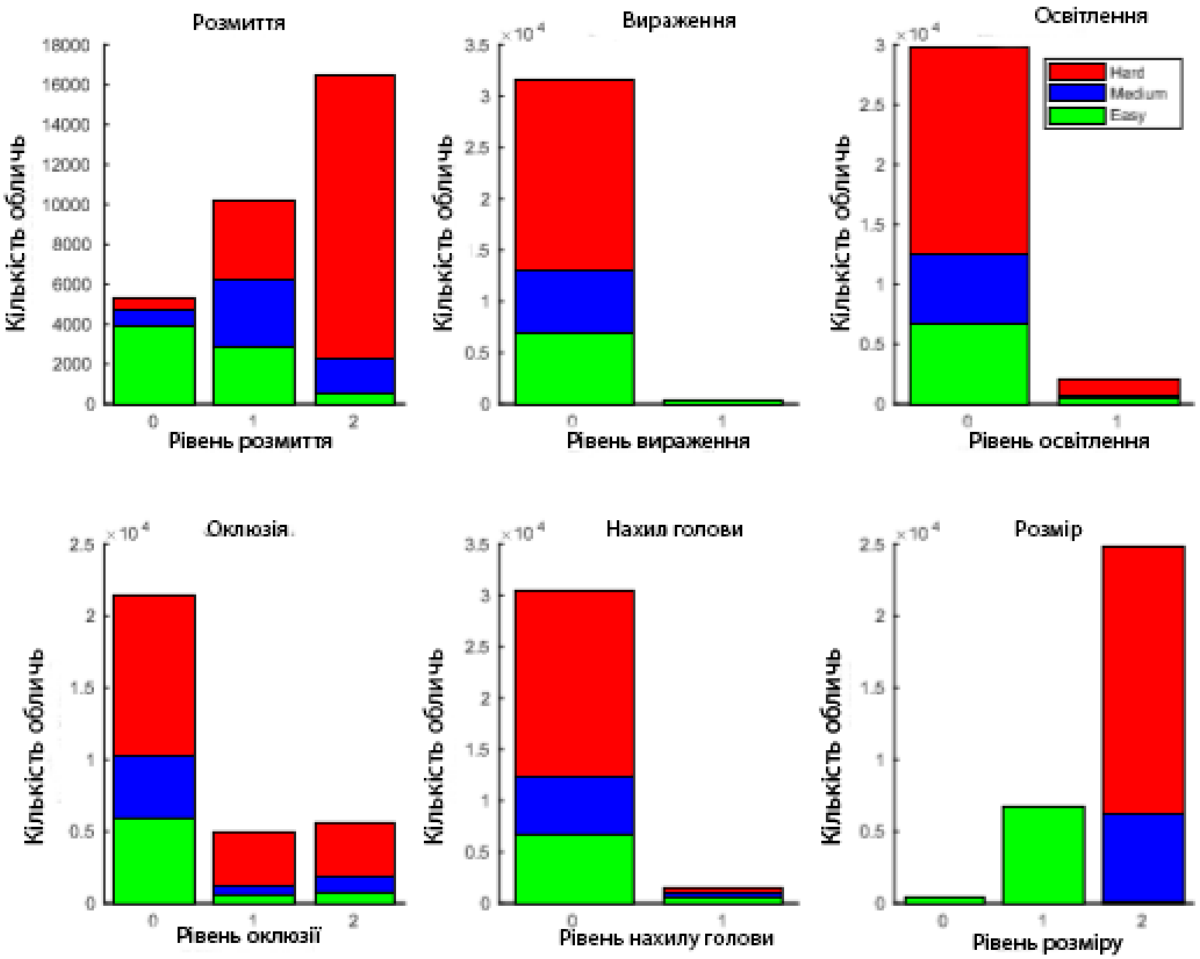


Рисунок 3.1 – Структура даних WIDER FACE

Файли ".mat" також містять основну опис для кожного обличчя. Основна інформація складається з прямокутних обмежувальних рамок, які позначаються як "x1, y1, w, h". x1 та y1 визначають положення верхнього лівого кута прямокутного обмежувального поля, x1 - це положення у пікселях з лівої частини зображення, а y1 - положення у пікселях з верхньої сторони зображення. w - ширина і h - висота зображення, в пікселях, зображення.

Для оцінки ефективності алгоритму розпізнавання обличь функція оцінки розраховує отримати для кожного зображення, що містить обличчя в наборі оцінок, перелік обмежувальних полів у тому ж форматі, що і основні дані.

3.1.2 Оцінка часу

Як зазначено в розділі 1.6, важливим є пошук того, чи можуть найсучасніші алгоритми працювати в обчислювальному середовищі заданої потужності Для цього необхідно порівняти час обчислення кожного алгоритму виявлення обличчя.

Час вимірювали під час тестів на валідаційному наборі WIDER FACE. Цей набір даних складається із зображень постійної висоти 1024 пікселів, але різної ширини з медіаною 754 пікселів. Через цю характеристику необхідно виміряти час виявлення для кожного кадру окремо, щоб можна було оцінити вплив розміру на час обчислення, одночасно маючи змогу обчислити усереднений час для всіх кадрів.

Для кожного зображення час вимірювався протягом усієї фази розпізнавання обличчя, тобто, час враховує усі операції попередньої обробки (зміни розміру) та остаточної обробки, але не враховує завантаження бібліотек чи мереж і збереження даних. Для алгоритмів, написаних на Python 2.7, можна легко отримати доступ як до процесора, так і до реального часу (через годинник функцій та час бібліотеки time відповідно), тоді як у C++ отримати доступ до часу процесора лише через функція clock бібліотеки ctime. При цьому час будє виражено чисельністю секунд для аналізу одного зображення, або як кількістю кадрів, які можна обробити за секунду (FPS), залежно від ситуації.

3.1.3 Тестування алгоритмів

VJ

Найвідомішою та найбільш використовуваною реалізацією VJ є бібліотека комп'ютерного зору , OpenCV. Перша важлива частина реалізації - це каскадний класифікатор. OpenCV пропонує ряд заздалегідь підготовлених класифікаторів , що зберігаються в публічно доступних файлах XML. Друга частина - алгоритм сам по собі. У OpenCV, алгоритм кодується, як в C++ і Python, в вигляді функції називається detectMultiScale

Серед цих параметрів особливий інтерес представляють чотири з них. По-перше, scaleFactor - параметр, який визначає, наскільки масштабується детектор сторінки на кожному кроці. До цього параметра відносяться параметри minSize та maxSize, які визначають відповідно початковий та кінцевий розмір детектора в процесі масштабування. Четвертий параметр називається minNeighbours і визначається як мінімальну кількість областей з виділеним обличчям, що перекриваються. Збільшуючи цей параметр, ми зменшуємо кількість помилкових позитивних рішень, видаляючи зайві виявлення. В роботі використовувалися такі значення цих параметрів:

• Cascadeclassifier: haarcascade frontalface alt2.xml

• scaleFactor : 1.3

• minNeighbors : 7

• minSize : 64x64

• maxSize : Default = Size of the image

Нарешті, реалізація цієї функції існує і є подібною як у C++, так і в Python, тому її було протовано на обох мовах.

Таблиця 3.1 - Список перевірених версій алгоритму VJ з різною мовою кодування та minSize

Ім'я	Мова кодування	масштаб фактор	minSize
VJ-C++-scale-1.3-min-10	C++	1.3	10x10
VJ-C++-scale-1.3-min-64	C++	1.3	64x64
VJ-python-scale-1.3-min-10	Python	1.3	10x10
VJ-python-scale-1.3-min-64	Python	1.3	64x64

На рисунку 3.2 показані криві ROC для цих чотирьох конфігурацій для легких, середніх та жорстких підмножин WIDER FACE, тоді як таблиця 3.2 містить кількість кадрів, які можна обробити за секунду, як для процесора, так і для реального часу.

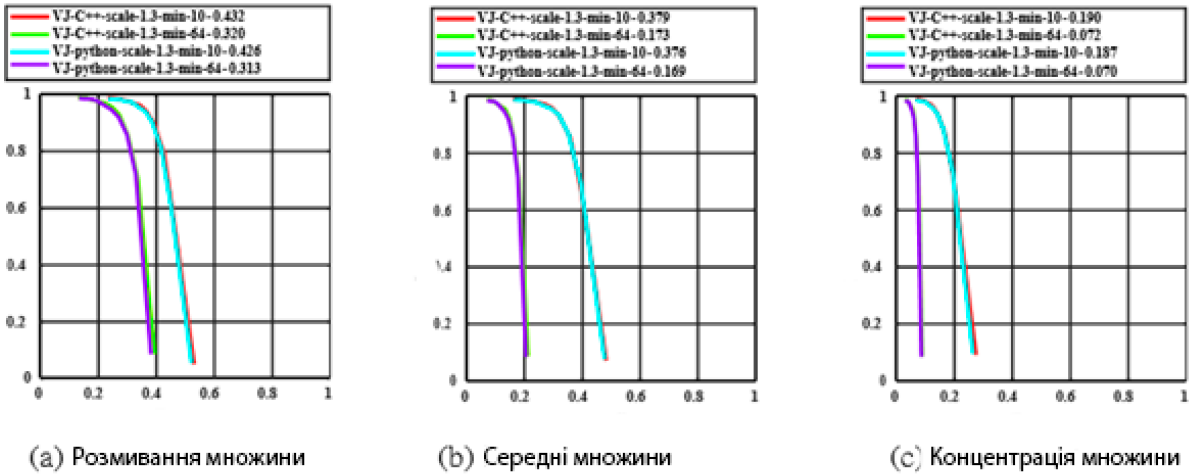


Рисунок 3.2 – ROC-криві, отримані за допомогою реалізації C++ та Python OpenCV алгоритму виявлення VJ з різними значеннями minSize на трьох підмножинах WIDER FACE.

Таблиця 3.2 - Медіани кількості кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, за допомогою реалізації C++ та Python OpenCV алгоритму виявлення обличчя VJ з різними значеннями minSize.

Алгоритм	FPS CPU	Справжній FPS
VJ-C++-scale-1.3-min-10	2.28	–
VJ-C++-scale-1.3-min-64	7.51	–
VJ-python-scale-1.3-min-10	2.56	35.98
VJ-python-scale-1.3-min-64	7.60	61.38

Перший висновок, яке ми можемо зробити, - це те, що зміна мови реалізації не впливає суттєво на продуктивність алгоритму, навіть незважаючи на те, що реалізація C++, як правило, працює краще. Однак, з точки зору ефективності обчислень, схоже, що реалізація python працює краще, ніж C++, покращуючи час процесора на 12,3% та 1,3%, для minScale = 10x10 та 64x64 відповідно. Ми можемо бачити, що чим менша minScale, тим більший розрив у часі обчислення між двома реалізаціями. Однак minScale впливає на дві реалізації однаково: коли вона зменшується, це означає що алгоритм може

спробувати виявити менші грані, їх ефективність поліпшується при збільшенні часу їх обчислення. Ми бачимо, що з точки зору FPS в режимі реального часу ми все ще досягаємо досить хорошого числа, коли використовуємо minScale = 10x10, в той час як ми можемо побачити різке поліпшення виступів, особливо для більш складних підгруп. В цілому ми можемо бачити, що за допомогою VJ ми можемо отримати хорошу точність.

Ще одна цікава характеристика в таблиці 3.2 полягає в тому, щоб побачити велику різницю між процесором та реальним часом, демонструє, що реалізація добре використовує багато потоковість.

Побачивши ці перші результати та високий рівень FPS, який можна було досягти навіть при невеликому значенні minSize, було вирішено зробити додатковий тест, встановивши scaleFactor на значення за замовчуванням, 1.1. Мета цього експерименту полягає в основному, щоб дізнатися, чи існує необхідність зменшення значення цього параметра. Оскільки реалізація Python була найшвидшою, а її продуктивність схожа на реалізацію C++, то протестувалися лише два додаткові параметри, наведені в таблиці 3.3, і порівняти їх з попередніми налаштуваннями python. Рисунок 3.2 та Таблиця 3.2 ілюструють це порівняння. Основний висновок полягає в тому, що спостерігається поліпшення продуктивності порівняно з попередніми параметрами, але що це покращення значно менше порівняно з результатом, отриманим при зменшенні значення minSize. Більше того, час обчислень погіршується. Дійсно, ми можемо побачити, що кількість FPS ділиться більш ніж на два як з точки зору процесора, так і в режимі реального часу.

Таблиця 3.3 - Список додаткових випробуваних версій алгоритму VJ, написаного на Python, із різною scaleFactor та minSize.

Ім'я	Мова кодування	масштаб фактор	minSize
VJ-python-scale-1.1-min-10	Python	1.1	10x10
VJ-python-scale-1.1-min-64	Python	1.1	64x64

Відповідно до цих результатів, я зупинюся лише на версії VJ-python-scale-1.3-min-10 для порівняння з іншими алгоритмами, оскільки вона забезпечує один з найвищих рівнів продуктивності з більш ніж розумною швидкістю.

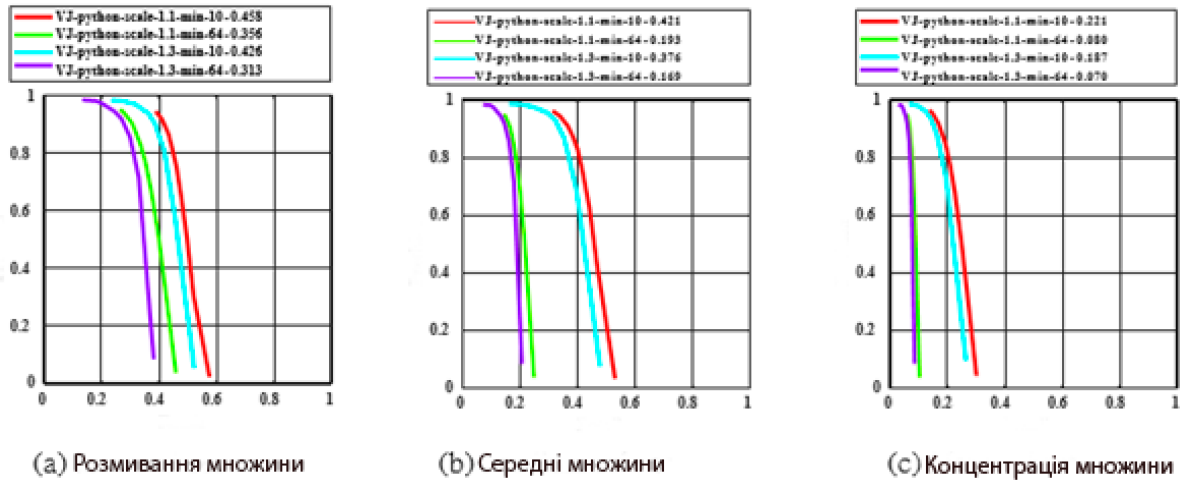


Рисунок 3.3 – ROC-криві отримані при реалізації Python OpenCV алгоритму виявлення обличчя VJ з різними значеннями minSize та scaleFactor на трьох підмножинах WIDER FACE

Таблиця 3.4 - Усереднена кількість кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, реалізацією Python OpenCV алгоритму виявлення VJ з різними значеннями minSize та scaleFactor

Алгоритм	FPS процесора	Справжній FPS
VJ-python-scale-1.1-min-10	1.08	14.39
VJ-python-scale-1.1-min-64	4.04	27.99
VJ-python-scale-1.3-min-10	2.56	35.98
VJ-python-scale-1.3-min-64	7.60	61.38

HOG

Цей алгоритм був реалізований у бібліотеці Dlib, який забезпечує реалізацію алгоритму в Python та C++. Цей алгоритм виявлення обличчя має лише два аргументи: зображення та параметр, який називається adjust threshold. У реалізації C++ використовується pyramid up для збільшення

розміру зображення. Цей метод використовується для того, щоб початковий детектор шукав лише обличчя розміром більше 80x80 пікселів. Однак використання цього методу уповільнює час обчислень.

У таблиці 3.5 наведені результати відклику точності для двох реалізацій на WIDER FACE, а в таблиці 3.6 показано FPS, отримані в режимі реального часу та процесора.

Таблиця 3.5 - Значення точності виклику, отримані за допомогою реалізацій Dlib Python та C++ алгоритму виявлення обличчя HOG.

Підмножина	Легко		Середній		Важкий	
Algorithm	Precision	Recall	Precision	Recall	Precision	Recall
HOG-python	0.9444	0.5160	0.9619	0.4153	0.9621	0.1744
HOG-C++	0.9611	0.3599	0.9611	0.1948	0.9611	0.0812

Таблиця 3.6 - Усереднена кількість кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, реалізаціями Dlib Python та C++ алгоритму виявлення обличчя HOG

Алгоритм	FPS процесора	Справжній FPS
HOG-python	3.22	3.07
HOG-C++	15.94	/

Аналіз таблиць показує, що так само, як VJ, HOG може досягти високої точності, але з низькою максимальною швидкістю відклику 51,6% у найпростішому тестовому наборі. Це значення потім зменшується для більш важких наборів. Точність не змінюється в різних наборах для реалізації C++.

Однак, аналіз часу обчислення показує, що реалізація C++ майже в 5 разів швидша, ніж реалізація Python. Останній досягає лише 3 FPS, що дуже повільно для практичної реалізації.

FRCNN

Реалізація, була написана на Python та використовує систему глибокого навчання Caffe. Як описано в оригінальній статті « Faster R-CNN», алгоритм виводить на кожне зображення список обмежувальних рамок з оцінкою достовірності від 0 до 1. Потім алгоритм тестування виконує два додаткові етапи. По-перше, він видаляє всі виявлення, пов’язані з оцінкою нижче певного порогу. Потім застосовується фільтрація NMS, що в ході дослідження приймало значення 0 (прибрати все, крім одного виявлення), 0,15 (за замовчуванням), 0,3, 0,5, 0,8 і 1 (нічого).

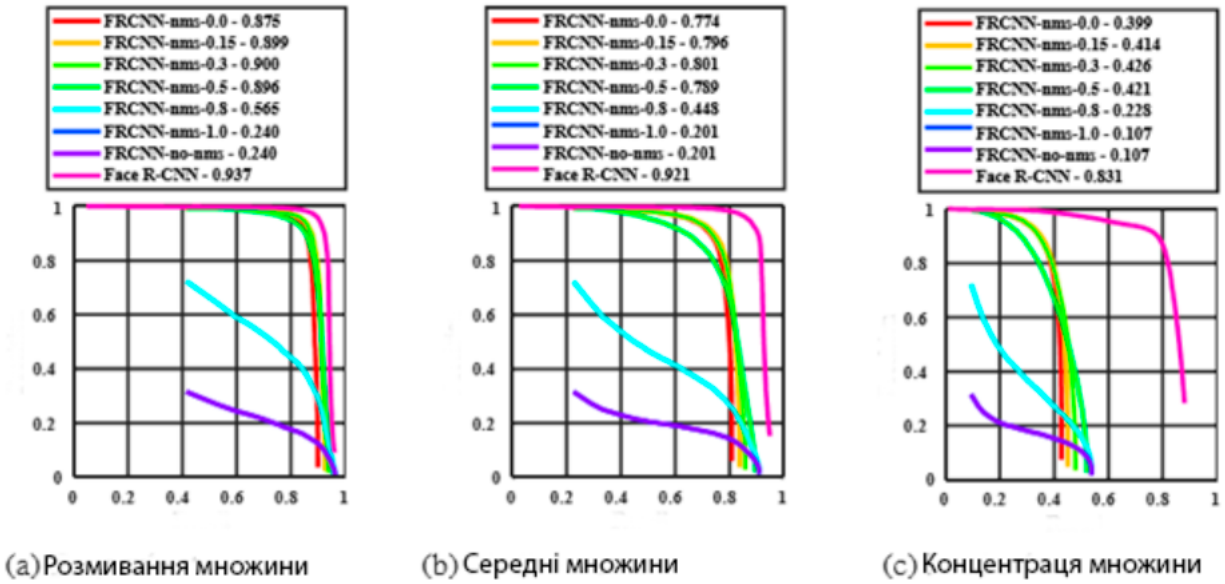


Рисунок 3.4 - Криві точності виклику, отримані за допомогою алгоритму FRCNN з використанням різних значень порогу NMS або відсутності NMS на трьох підмножинах WIDER FACE

Основна інформація, яку слід зібрати з рис. 3.4, полягає в тому, що зміна порогового значення NMS не впливає на час обчислення, ймовірно, тому, що застосування NMS обчислювання незначне порівняно з фазою виявлення. Для FRCNN ця фаза виявлення залишається досить ефективною, оскільки ми можемо отримати швидкість, що перевищує 15 FPS. Нарешті, ми можемо

побачити ефект паралельної обробки, дивлячись на різницю між процесором та FPS у режимі реального часу. Ця різниця менша, ніж у VJ, але це може бути пояснено тим, що у випадку FRCNN обчислення відбуваються також на GPU. Тому цей час розглядається лише як реальний час, що призводить до зменшення кількості FPS у режимі реального часу порівняно з FPS часу CPU.

На рисунку 3.4 показано, що FRCNN може досягти набагато вищої швидкості виклику, ніж два попередньо протестованих методу з більш ніж 90% на легшій підмножині для деяких значень порогу NMS. Переглядаючи варіацію порогу NMS, ми можемо побачити, що для трьох підмножин поведінка однакова. Коли NMS поріг збільшується, на рівнях з низьким рівнем відклику, точність зменшується, а при низькому рівні точності, збільшується. Крім того, ми можемо бачити, що аж до порогового значення 0,5, збільшення відклику ще компенсує зменшення точності, із середньою точністю (*AP*) перебування в розумних межах, але, що збільшення порогу погіршує характеристики. При цьому продуктивність випробуваних алгоритмів значно зменшується при збільшенні складності набору, криві Face R-CNN є досить послідовними і призводять до дуже хороших показників навіть на найскладнішій підмножині. Нарешті, якщо ми проаналізуємо *AP* як міру ефективності, то значення 0,3 для порогу NMS виявляється оптимальним.

Таблиця 3.7 - Усереднена кількість кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, алгоритмом FRCNN

Алгоритм	FPS процесора	Справжній FPS
FRCNN-nms-0.0	8.83	15.69
FRCNN-nms-0.15	8.82	15.59
FRCNN-nms-0.3	8.84	15.69
FRCNN-nms-0.5	8.84	15.69
FRCNN-nms-0.8	8.83	15.62
FRCNN-nms-1.0	8.82	15.51
FRCNN-no-nms	8.83	15.65

SFD

Реалізація SFD написана на Python і використовує Caffe. У частині тестування представлена попередньо підготовлена мережа VGG [79] на навчальному наборі WIDER FACE.

Що стосується самої реалізації тесту, то у оригінальному сценарії використовуються три варіанти (позначені det0, det1 та det23 відповідно). Останній набір операцій застосовується до виявлень через функцію, яку називають bbox. Мета цієї функції - об'єднати виявлення, які занадто сильно перекриваються, аналогічно NMS. Відмінність від NMS полягає в тому, що обмежувальні поля, які занадто сильно перекриваються вікном з максимальним обмеженням достовірності, не просто видаляються, але всі обмежувальні поля замінюються обмежувальним вікном, координати якого є середньозваженим значенням координат перекриваючих обмежувальних полів, аналогічно до того, що було описано в роботі VJ.

На рисунку 3.5 показано результат для чотирьох випадків, викликаних до цього, плюс комбінації першого та другого наборів (det0, det1). BBOX поріг був на 0,3 для всіх цих експериментів. Тоді ми бачимо, що якщо об'єднати всі варіації разом (тобто деталізувати), ми отримаємо криву, яка має досить непоганий виклик для низьких значень точності, але що при зменшенні швидкості виклику точність підвищується, а потім знову починає знижуватися. Особливо це видно на жорсткій підмножині. Така поведінка означає, що коли ми збільшуємо поріг прийняття рішення, кількість TP зменшується швидше, ніж кількість FP. Інший спосіб сказати, що при більш достовірному відношенню TP до FP менше, ніж при більш низьких рівнях достовірності. Така поведінка вже була дивною, але все ж пояснювальною.

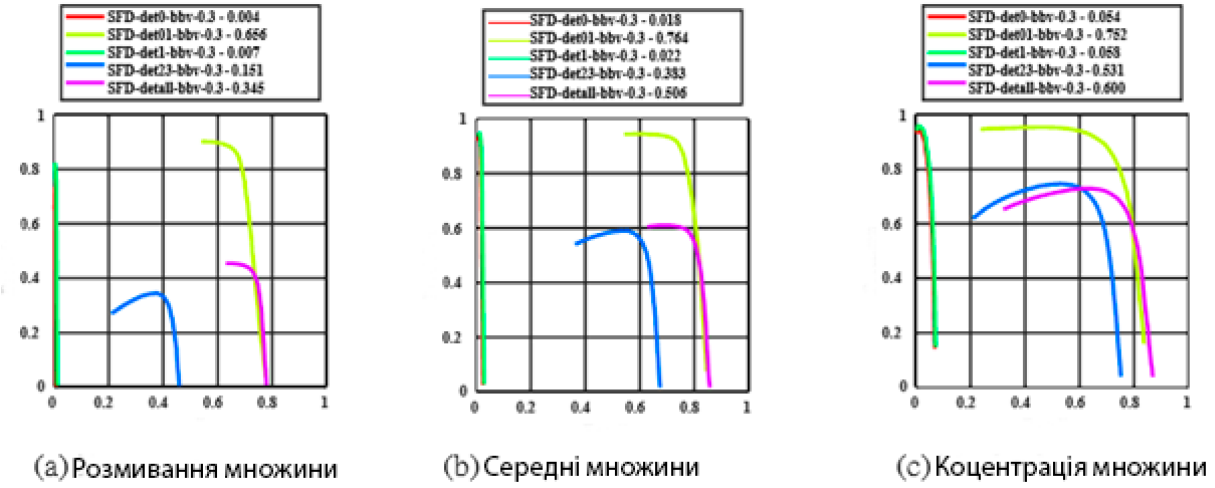


Рисунок 3.5 - ROC-Криві, отримані з варіаціями, використовуючи функцію bbox алгоритму SFD на трьох підмножинах WIDER FACE

Для того, щоб можна було порівняти SFD і інші методи замінемо bbox методом NMS. Результати показані на рисунку 3.6.

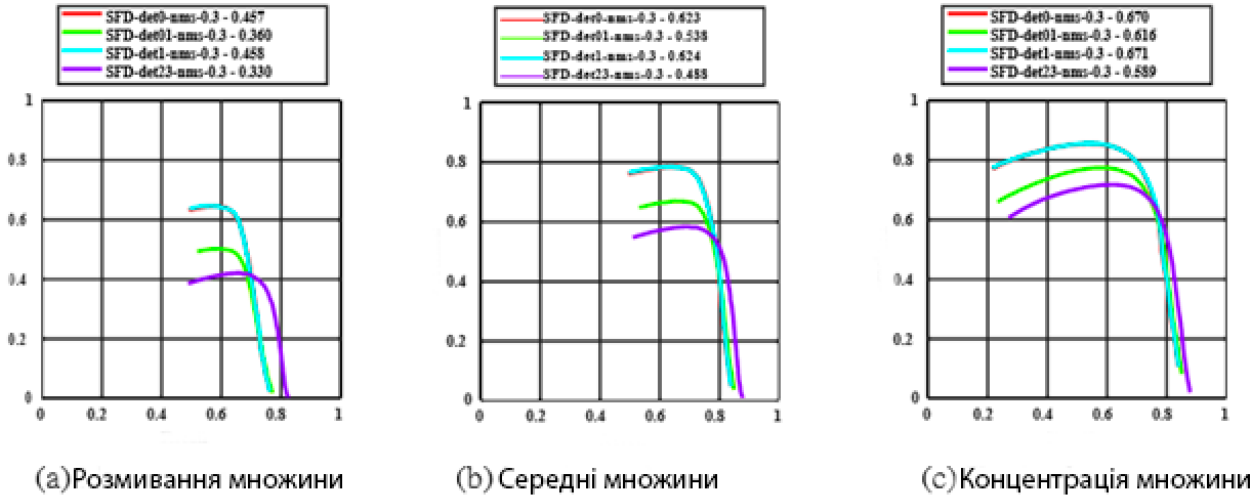


Рисунок 3.6 – ROC- криві, отримані з варіаціями, використовуючи NMS алгоритму SFD на трьох підмножинах WIDER FACE

Дійсно, ми бачимо, що поєднання цих двох варіантів тепер призводить до гіршої продуктивності, ніж це відображала функція bbox.

Для закінчення цього алгоритму в таблиці 3.8 показано час обчислення на кадр та FPS для різної версії алгоритму. У таблиці відображаються результати лише bbox, але результати, отримані за NMS, майже однакові, з максимальною різницею 0,2 FPS на користь NMS.

Таблиця 3.8 – Усередненна кількість кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, за варіаціями, використовуючи голосування bbox алгоритму SFD.

Алгоритм	FPS процесора	Справжній FPS
SFD-det0-bbv-0.3	10.17	10.17
SFD-det1-bbv-0.3	9.95	9.95
SFD-det01-bbv-0.3	5.04	5.04
SFD-det23-bbv-0.3	0.88	1.08
SFD-detall-bbv-0.3	0.76	0.91

SSH

Реалізація протоколу тестування дуже схожа на протокол FRCNN. Дійсно, в цьому випадку також можна використовувати NMS з різними пороговими значеннями, а також змінювати рівень достовірності, при якому ми не хочемо зберігати виявлення. Що стосується FRCNN, ми встановимо рівень достовірності до 0 та перевіримо значення 0, 0,15, 0,3, 0,5, 0,8 та 1 для порогу NMS. На додаток до цього, існує можливість зміни масштабу вхідних зображень у фіксованому масштабі або за допомогою піраміди масштабу. Я вирішив вивчити лише перший варіант, оскільки він вважався найкращим компромісом між швидкістю та точністю в папері SSH.

Аналізуючи рисунок 3.7, ми можемо побачити, що поведінка кривих, що змінюються залежно від порогу NMS, дуже схожа на випадок FRCNN (тобто, коли значення порогового значення збільшується, при низькому виклику точність зменшується і при низькій точності, збільшується виклик) найкраще порогове значення в цьому випадку становить 0,5, як випливає із значення AP. Однак головна відмінність полягає в тому, що продуктивність для SSH краща,

ніж для FRCNN, і знижується набагато менш швидко при тестуванні на більш жорсткі підмножини. На жаль, це за замовчуванням більш повільний час обчислень, що йде від 8,8 FPS до 4 FPS за час процесора і від 15,6 до 6,7 FPS в режимі реального часу, як показано в таблиці 3.14. Більше того, ми можемо ще раз побачити, що порогове значення NMS, схоже, не впливає на час обчислення і що багато потокова робота працює нормально.

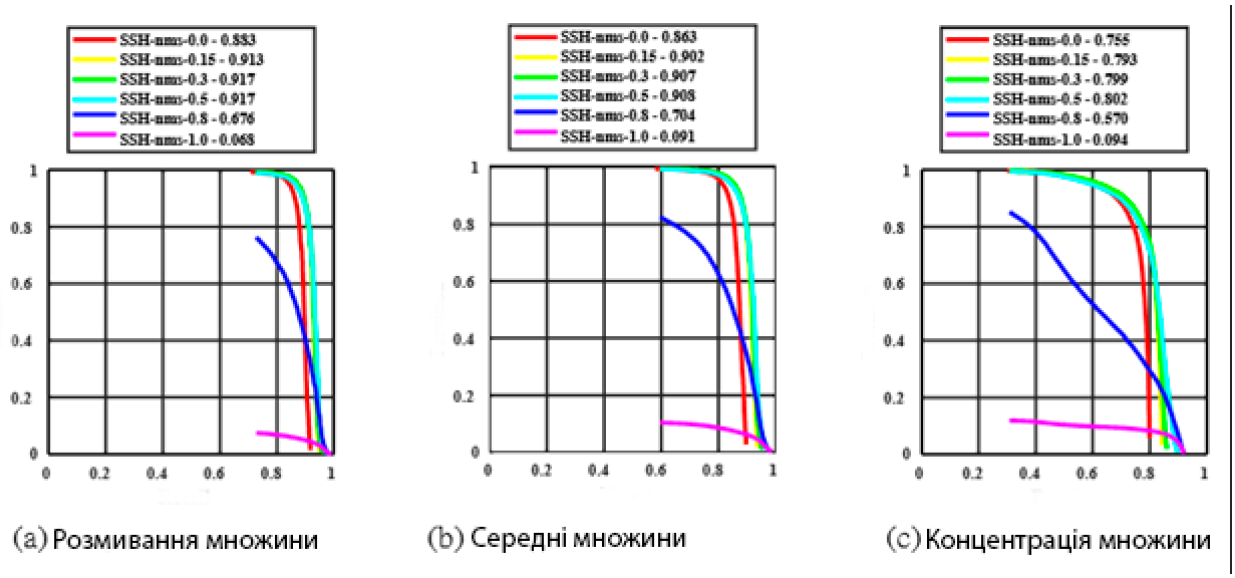


Рисунок 3.7 - Криві точності виклику, отримані за допомогою алгоритму SSH з використанням різних значень порогу NMS або відсутності NMS, на трьох підмножинах WIDER FACE

Таблиця 3.9 - Усередненна кількість кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, за допомогою алгоритму SSH, використовуючи різні значення порогу NMS або відсутність NMS

Алгоритм	FPS процесора	Справжній FPS
SSH-nms-0.0	4.07	6.68
SSH-nms-0.15	4.07	6.67
SSH-nms-0.3	4.06	6.68
SSH-nms-0.5	4.09	6.68
SSH-nms-0.8	4.06	6.67
SSH-nms-1.0	4.06	6.66

3.1.4 Аналіз результатів

Перш ніж проаналізувати вплив атрибутів зображень обличчя, на узагальнені результати різних алгоритмів, залишимо лише по одному найкращому показнику кожного методу детектування обличчя.

З точки зору точності рішень, чіткий переможець - SSH, який єдиний, кому вдається досягти значень відклику, що перевищують 50% для кожної з трьох підмножин. FRCNN характеризується непоганою точністю на простому та середньому наборі, але втрачає її на найскладніших наборах даних. Що стосується VJ та HOG, то їх точність майже однакові, але нижче FRCNN та SSH.

Таблиця 3.10 - Кількість кадрів, що надходять від WIDER FACE, які можуть оброблятися за секунду, в процесорі та в режимі реального часу, з найкращими версіями перевірених алгоритмів.

Алгоритм	FPS процесора	Справжній FPS
VJ-python-scale-1.3-min-10	2.56	35.98
HOG-C++	15.94	/
FRCNN-nms-0.3	8.84	15.69
SSH-nms-0.5	4.09	6.68

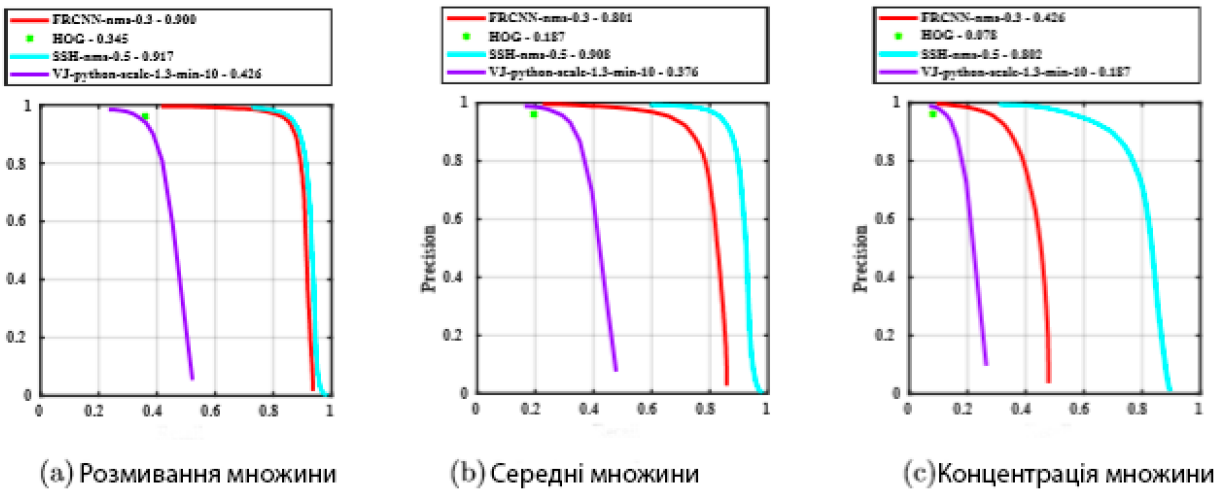


Рисунок 3.8 - Криві точності детектору, отримані найкращими версіями перевірених алгоритмів на трьох підмножинах WIDER FACE.

Щодо оперативності в реальному часі, то будемо вважати кращим алгоритм з меншою кількістю кадрів в секунду. Оскільки для HOG ми не маємо доступу до даних у режимі реального часу, то зосередимось лише на FRCNN та SSH. Перший алгоритм у більш ніж у два рази швидший за другий. Ця різниця підтримується на тому , що SSH складніше використовувати GPU при роботі. Дійсно, хоча FRCNN використовує лише близько 2000 МБ пам'яті GPU , SSH використовує майже 4500 МБ. Це показує, що SSH є набагато більш складною моделлю , яка потребує більше часу на обчислення

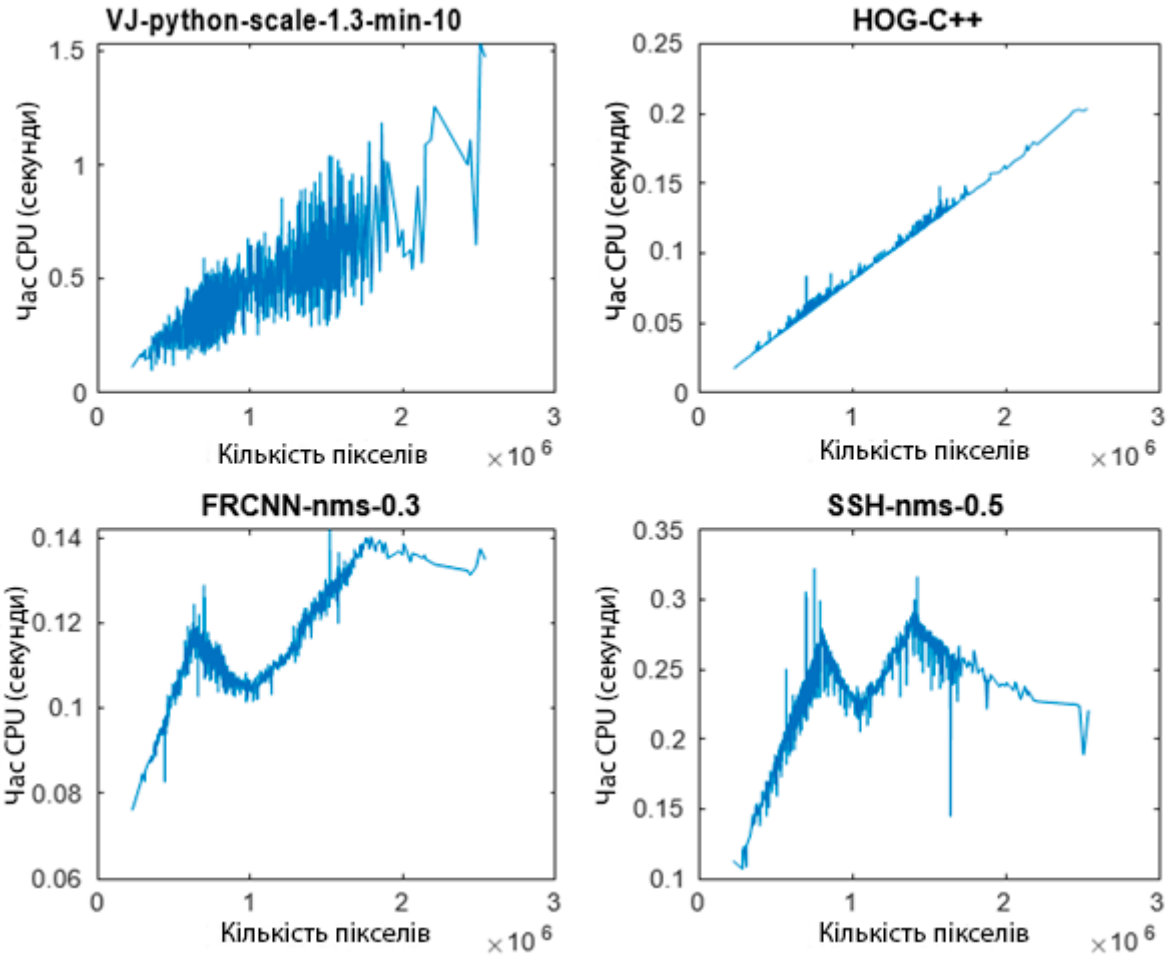


Рисунок 3.9 - Зростання часу обчислення процесора на одне зображення з кількістю пікселів на зображення для 4 тестованих алгоритмів.

Нарешті, на рисунку 3.8 для 4 алгоритмів показано еволюцію часу процесора з кількістю пікселів на зображення. Навіть якщо дані є досить з великим шумом на зображенні, ми можемо побачити, що і для VJ, і для HOG

час, здається, лінійно збільшується із кількістю пікселів. Однак для FRCNN та SSH відношення далеко не лінійне. Це явище насправді пов'язане з тим, що зображення переробляються перед подачею в мережі. Однак в цілому, здається, час збільшується з кількістю пікселів, навіть якщо в цих двох випадках важче екстраполювати великі значення.

3.2 Розпізнавання обличчя

3.2.1 Протокол оцінювання

Як пояснено в підрозділі 1.8, що потрібно зробити тестування алгоритмів розпізнавання обличчя на еталоні MegaFace. Зараз ми це все детально опишемо більш докладно, як ці тести проводилися за допомогою інструментів, наданих цим еталоном.

Код оцінки MegaFace дозволяє перевірити ідентифікацію обличь із зображень двох базових наборів, FaceScrub та FGNet, проти алфавіту класів розпізнавання, що містить різну кількість дистрибуторів (від 10 до 1000000). Як було зазначено раніше, оскільки нас не цікавить розпізнавання обличчя за різними віковими варіантами, ми зупинюся лише на наборі даних FaceScrub. Важливо уточнити, що для „ефективності” [47] протокол оцінювання використовує лише підмножину FaceScrub, включаючи 80 осіб (40 жінок та 40 чоловіків) з 50 зображень кожний.

Нарешті, нам потрібно додати, що і для MegaFace (MF), і для FaceScrub (FS) надається інформація про обмежувальні рамки, що дозволяє легко обрізати кожне зображення, щоб зберегти лише обличчя. Потім це обрізане обличчя проходить через деяку попередню обробку (наприклад, вирівнювання), а потім витягування.

3.2.2 Оцінка часу

Існує два основні етапи розпізнавання обличчя, які необхідно проаналізувати окремо за часом: створення функцій (включаючи всі етапи

попередньої обробки) та класифікація ознак (отримана за допомогою класифікації найближчих сусідів). Важливо обчислити два обчислення часу окремо, тому що ці дві задачі легко можна виконувати паралельно. Більше того, ефективність цих двох фаз не менш важлива для оцінки. Дійсно, хоча ми можемо вважати, що генерація можливостей є найбільш змінною за часом обчислення, тривалість створених функцій буде лінійно впливати на час обчислення другої фази, який може стати значним, коли розмір галереї руйнується.

Для першого етапу вимірювали час для кожного зображення набору даних MegaFace. Ми не обчислювали це за зображеннями FaceScrub, оскільки це було б надлишково. Тут важливо зауважити, що кількість зображень дорівнює кількості обличь. Однак, якщо розглянути повний конвеєр розпізнавання обличь, кожне зображення, можливо, містить кілька граней. Щоб уникнути плутанини з FPS, що використовується метрикою часу для алгоритмів виявлення обличчя, ми буду використовувати для розпізнавання обличчя термін FaPS, посилаючись на кількість оброблених обличь за секунду. Цей час на обличчя враховуватиме вирівнювання та особливості вилучення, але не обрізання, оскільки це не залежить від обраного алгоритму.

Для другої фази не вдалося отримати час обчислення для зображення, оскільки код тестування був доступний лише у вигляді виконуваного файлу, що працює над усім базовим набором та алфавіту класів розпізнавання, використовуваним для тесту, і його неможливо було змінити. Єдиною можливою диференціацією було обчислення часу для різних розмірів алфавіту класів розпізнавання. Однак, як код тестування був написаний на Python, ми мали доступ як до реального, так і до процесорного часу. Більше того, оскільки цей час залежить лише від розміру видобутого функціонального вектора, вони будуть однаковими для кожного з випробуваних алгоритмів, 3 варіантів. Тому ми проаналізували його лише у розділі загальних результатів.

Що стосується розпізнавання обличь, ми виміряли процесор і реальний час (коли це було можливо), плюс використання процесора та графічного процесора, використовуючи ті самі функції та команду.

3.2.3 Тестування алгоритмів

OpenFace

Бібліотека OpenFace надає код Python для навчання та тестування з попередньо перевіреними мережами. Серед перевірених мереж найкраща ефективність на LFW - це "nn4.small2.v1", яка базується на спрощеній версії Inception.

Зараз будемо перевіряти варіацію по алгоритму:

• OpenFace: OpenFace без вирівнювання

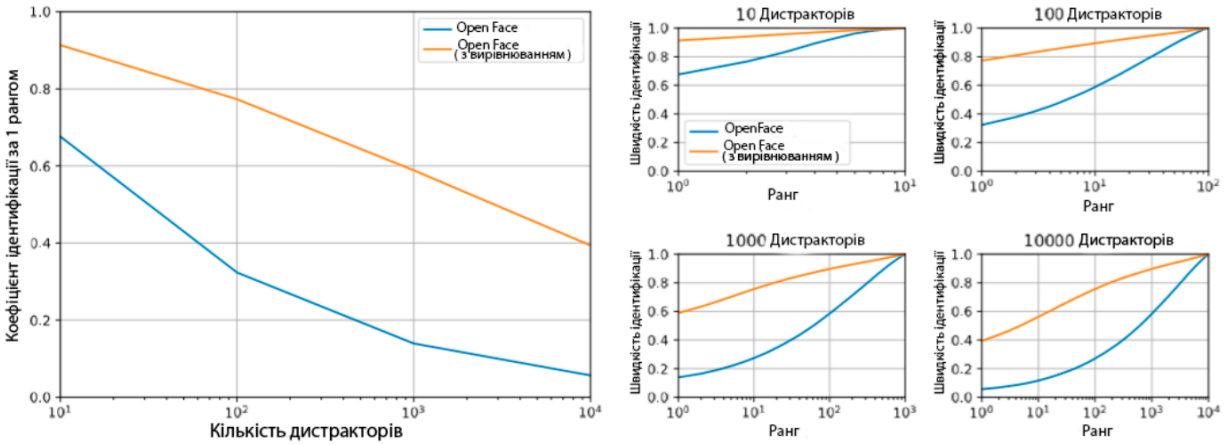
• OpenFace : OpenFace з вирівнюванням

На рисунку 3.10 показано (a) продуктивність рангу-1 для різних номерів дистракторів, а в (b), воно показує, як розвивається ідентифікаційна ефективність з рангом для цих 4 різних номерів дистракторів. Як і очікувалося, продуктивність зменшується з кількістю дистракторів, в той час як вона збільшується при збільшенні рангу. Здається, дистрактор значно покращить показники, особливо якщо ми враховуємо більшу кількість дистракторів Дійсно, ми можемо бачити, що з 10 дистракторів продуктивність OpenFace без вирівнювання є прийнятною із рівнем ідентифікації при ранзі-1 близько 70%. Однак ця величина падає дуже швидко і сягає менше 10% для 10 000 дистракторів. За трьома найбільшими числами дистракторних факторів вирівнювання подвоює коефіцієнт ідентифікації, досягаючи приблизно 60% для 1000 дистракторів ще 40% для 10 000 дистракторів. Якщо ми подивимось на другий графік, ми фактично побачимо, що для обох варіантів швидкість ідентифікації зростає досить повільно з рангом.

Щодо часу обчислень, Таблиця 3.11 показує, що вирівнювання змушує кількість FaPS зменшитись досить різко до всього 25 обличь за секунду в реальному часі. Гірша швидкість ідентифікації, отримана без вирівнювання,

компенсується збільшенням обчислень у 4-5 разів. Це збільшення часу на обчислення ще більше, якщо врахувати час процесора. Це, безумовно, означає, що вирівнювання здійснюється за допомогою процесора, а не з реалізацією GPU. В документації на OpenFace цього немає чіткого пояснення. Однак, дивлячись на активність GPU з вирівнюванням чи ні, немає різниці.

Оскільки продуктивність OpenFace без вирівнювання занадто погана для практичного застосування, ми зберігатимемо версію OpenFace для подальшого аналізу, хоча це набагато повільніше.



(a) Ідентифікаційний показник 1-го рівня та кількість дистракторів

(b) криві CMC для різної кількості дистракторів.

Рисунок 3.10 - Виконання, отримані за алгоритмом OpenFace, з вирівнюванням і без нього, на MegaFace. (а) показує зріст швидкості ідентифікації 1-го рангу з кількістю дистриб'юторів у галереї, а (б) показує криві CMC для цієї різної кількості дистракторів.

Таблиця 3.11 – Усереднена кількість обличь, що надходять від MegaFace, які можуть оброблятися в секунду, в процесорі та в режимі реального часу, за допомогою алгоритму OpenFace з вирівнюванням і без нього.

Алгоритм	FaPS процесора	Справжні FaPS
OpenFace	230.47	110.81
OpenFace (Align)	29.89	24.66

Dlib-R

Бібліотека Dlib містить перевірену модель на основі спрощеної версії коду тестування ResNet-34 та C++ для Dlib-R у вигляді прикладу, що дозволяє виконувати витяг функцій на одному зображенні, що містить кілька граней. Вирівнювання також можна виконати за допомогою алгоритму вирівнювання обличчя Dlib, викликаного OpenFace, але за допомогою передбачуваного формату 5 обличь. Тоді точність розпізнавання обличь очевидно може бути покращена, якщо для створення дескрипторів обличчя використовується "тремтіння". Функція, яка дозволяє додати тремтіння до процесу, насправді складає лише 100 копій вирівнюваного зображення, «все трохи висвітлене, якщо його збільшити, повернути та перекласти трохи по-іншому. Вони також довільно відображаються зліва направо ». Потім розпізнавання обличчя застосовується до кожного з цих зображень. Цей процес, на жаль, уповільнює процес розпізнавання.

Оскільки у нас є можливість застосувати вирівнювання це призводить до 4 різних можливостей тестування:

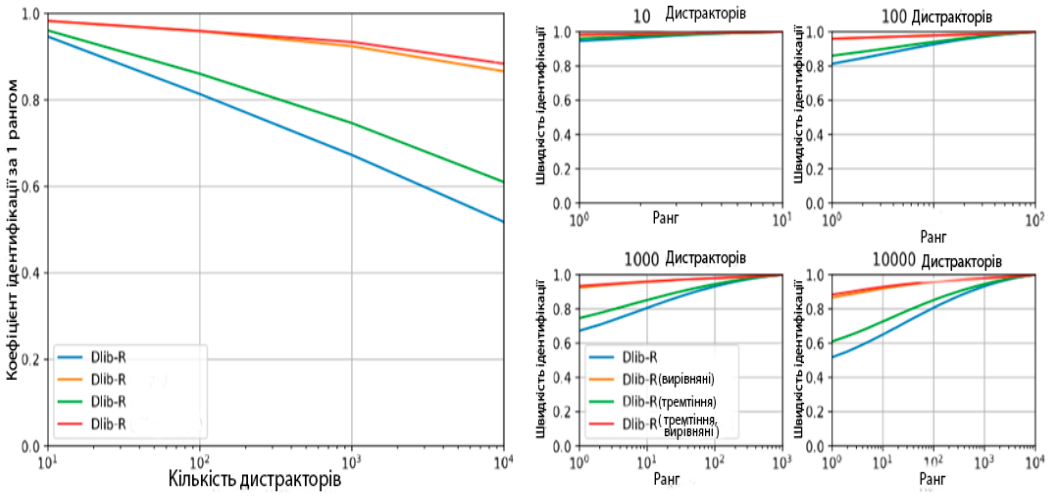
- Dlib-R: Dlib-R без тремтіння чи вирівнювання
- Dlib-R: Dlib-R з вирівнюванням і без тремтіння
- Dlib-R :Dlib-R без вирівнювання та з тремтінням
- Dlib-R: Dlib-R з вирівнюванням і тремтінням

На рисунку 3.11 видно, що для невеликої кількості дистракторів різниці в продуктивності 4-х варіацій дуже малі і вони збільшуються із цим числом. Якщо ми порівнюємо переваги вирівнювання і джиттера, ми можемо

побачити, що вирівнювання, здається, забезпечує великі поліпшення MENT. Дійсно, дивлячись на найбільшу кількість дистракторів, ми можемо побачити, що порівняно з базовим рівнем реалізації забезпечується покращення на 35%, тоді як тремтіння забезпечує лише трохи менше 10% покращення. Крім того, поєднуючи ці два, ми бачимо, що ми отримуємо лише 1% від рівня ідентифікації, порівняно з тим, коли ми використовуємо лише вирівнювання.

Поширеність вирівнювання додатково мотивується часом обчислень. Дійсно, в таблиці 3.26 ми бачимо, що і вирівнювання, і тремтіння зменшують кількість обличь, які можна обробити за секунду, але це тремтіння має набагато більш грубий вплив. Навпаки, порівняно з OpenFace, вирівнювання значно менше впливає на базовий час обчислення, зменшуючи його лише на коефіцієнт 2. Це може бути викликано тим, що використовується орієнтирна модель 5, а не орієнтир 68, або це може бути походять від того, що Dlib-R кодується в C++, тоді як OpenFace пишеться в Python. У нас не було часу досліджувати це більш глибоко.

У відповідності з цими результатами, ми будемо продовжувати подальший аналіз по Dlib-R версії.



(a) Ідентифікаційний показник 1-го рівня та кількість дистракторів

(b) криві CMC для різної кількості дистракторів

Рисунок 3.11 - Результати, отримані за алгоритмом Dlib-R, з і без вирівнювання, з і без тремтіння, на MegaFace

Таблиця 3.12 - Медіани числа граней, виходячи з MegaFace, які можуть бути оброблені за секунду, в процесорі і режим реального часу, з використанням алгоритмом Dlib-R, з і без вирівнювання, з і без тремтіння.

Алгоритм	FaPS процесора
Dlib-R	457.46
Dlib-R (Align)	246.67
Dlib-R (Jitter)	14.14
Dlib-R (Align, Jitter)	13.73

ArcFace

Реалізація ArcFace була випущена з проектом під назвою "InsightFace", який називається "проектом 2D та 3D аналізу обличчя" і є частиною сховища GitHub "Deep Insight" проведення багатьох проектів навколо глибокого навчання. Точніше, в цьому проекті вони "надають дані про навчання, налаштування мережі та дизайн втрат для глибокого розпізнавання обличчя". Дані тренінгу складаються з "нормалізованого" набору даних MS-Celeb-1M [33] та VGG2 набору даних [12]. Базова мережа включає в себе кілька конструкцій, включаючи ResNet, реалізовані за допомогою бібліотеки MXNet, і ряд функцій втрат тестуються, включаючи втрату триплет і "ArcFace".

InsightFace надає код навчання та тестування, а також перевірену мережу. Точніше, надаються LResNet50E-IR та LResNet34E-IR. Однак другий доступний лише через Baidu Drive, і для його отримання потрібен обліковий запис (якого у мене немає). Тестовий випадок дозволяє генерувати функції для FaceScrub та MegaFace, застосовуючи вирівнювання заздалегідь. Тому ми вирішили протестувати ArcFace з вирівнюванням і без нього. Основна відмінність полягає в тому, що точний процес вирівнювання, який використовується ArcFace, визначити трохи складніше, ніж OpenFace або Dlib-R. Дійсно, бібліотека ArcFace надає низку функцій для вирівнювання різних наборів даних, на яких вона може бути протестована. Одна з реалізацій використовувала ту саму техніку вирівнювання, що і OpenFace, і, здається, є

найпростішою у використанні, тому ми продовжували цей вибір. Два випробувані варіанти ми назвали так:

• ArcFace : ArcFace без вирівнювання

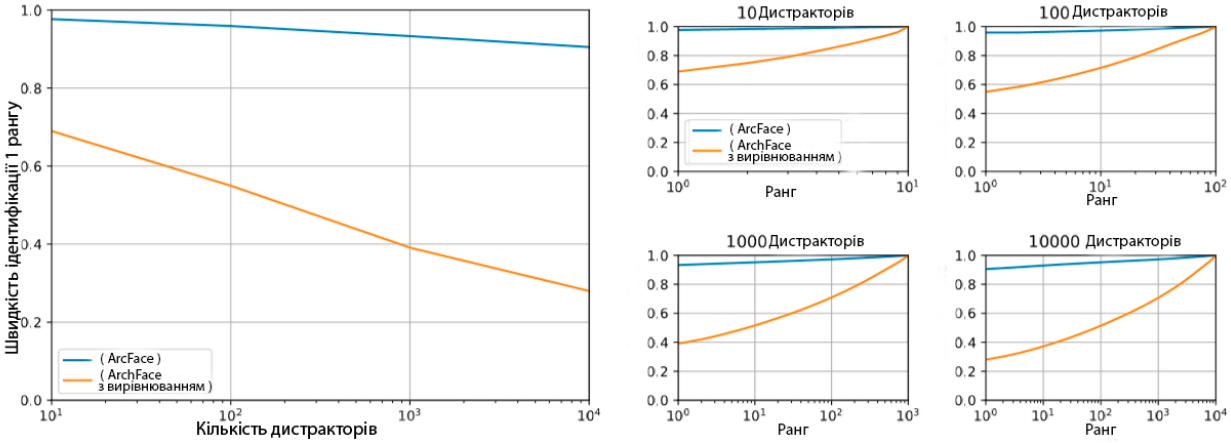
• ArcFace : ArcFace з вирівнюванням

Переглядаючи рисунок 3.12, ми бачимо, що несподівано використання вирівнювання негативно впливає на швидкість ідентифікації. Це повинно бути пов'язано з тим, що техніка вирівнювання, яка використовується в даному випадку, не підходить для наступного етапу вилучення ознак, що робить розподіл вхідного тестування занадто відмінним від розподілу вхідних тренувань. Навпаки, не використовуючи жодної техніки вирівнювання, ми отримуємо напрочуд гарні результати навіть для найбільшого розміру заданих дистракторів.

Таблиця 3.13 – Усереднена кількість осіб, що надходять від MegaFace, які можуть оброблятися в секунду, в процесорі та в режимі реального часу, за допомогою алгоритму ArcFace, з вирівнюванням і без нього

Алгоритм	FaPS процесора	Справжні FaPS
ArcFace	50.98	60.35
ArcFace (Align)	23.14	24.53

З точки зору часу на обчислення, не дивуючись, вирівнювання має негативний ефект. Це поєднання з раніше згаданим фактом змушує зберігати версію без узгодження. Звичайно, в майбутній роботі може бути цікаво вивчити інші методи вирівнювання, щоб побачити, чи покращують вони показники.



(a) Ідентифікаційний показник 1-го рівня та кількість відволікаючих

b) Криві CMC для різної кількості дистракторів

Рисунок 3.12 - Результати, отримані за алгоритмом ArcFace, з вирівнюванням і без нього, на MegaFace. (a) показує зростання швидкості ідентифікації 1-го рангу з кількістю дистракторів у галереї, а (b) показує криві CMC для цієї різної кількості дистракторів.

3.2.4 Аналіз результатів

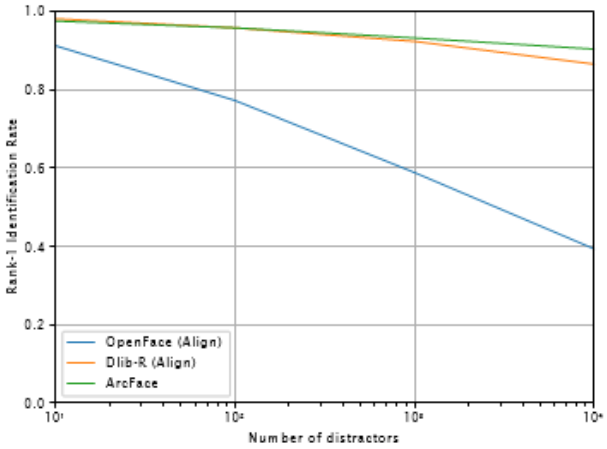
На рисунку 3.13 показані подібні графіки, як у трьох попередніх підрозділах, але зараз порівнюються найкращі версії кожного алгоритму розпізнавання обличь. Основне зауваження, яке можна зробити, - це те, що Dlib-R і ArcFace на один рівень вище OpenFace. Два колишніх алгоритми мають аналогічний рівень продуктивності, коли Dlib-R є дещо кращим для найменшої кількості відволікаючих факторів, а ArcFace виходить назустріч, коли ця кількість збільшується.

Однак якщо ми подивимось на час, з таблиці 3.14 показано, що час процесора Dlib-R працює набагато краще. Це може бути пояснено тим фактом, що вона написана на C++ , а не в Python і що це є алгоритм , який використовує найменш GPU пам'яті.

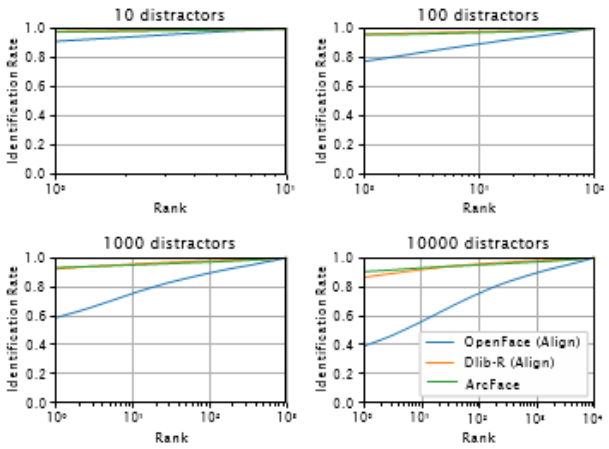
Таблиця 3.14 - Усереднена кількість осіб, що надходять від MegaFace, які можуть оброблятися в секунду, в процесорі та в режимі реального часу, з найкращими версіями трьох перевірених алгоритмів розпізнавання обличь

Алгоритм	FaPS процесора	Справжні FaPS
OpenFace (Align)	29.88	24.66
Dlib-R (Align)	246.67	/
ArcFace	50.98	60.35

Зараз ми коротко проаналізуємо час обчислення для етапу класифікації. Оскільки цей час залежить лише від розміру представлення функції, ми будемо посилатися на алгоритм за їх назвою без характеристик. Ці розміри відображені в таблиці 3.15. Ми можемо бачити, що Dlib-R і OpenFace генерують однакову кількість функцій на обличчя, тоді як InsightFace генерує в чотири рази більше.



(a) Ідентифікаційний показник 1-го рівня та кількість відволікаючих



(b) Криві CMC для різної кількості дистракторів

Рисунок 3.13- Результати, отримані за найкращими версіями 3 перевірених алгоритмів розпізнавання обличчя на MegaFace. (a) показує зростання швидкості ідентифікації 1-го рангу з кількістю дистракторів у галереї, в той час як (b) показує криві CMC для цієї різної кількості дистракторів.

На рисунку 3.14 показано зростання часу класифікації з кількістю використовуваних дистракторів. Ми можемо бачити, що, як очікується, для

ArcFace час більший. Ми також можемо побачити, що, здається, є невелика різниця між Dlib-R і OpenFace, яка неочікувана. Для кращої міри ми, можливо, мали б повторити експеримент чим ми і зайнялися. Якщо проаналізувати зростання часу з кількістю дистракторів, то схоже, у нас є лінійне відношення. Нарешті, з цього графіка ми також можемо приблизно зробити висновок про час, необхідний для визначення одного зображення. Ми знаємо, що загалом визначаються 4000 базових осіб. Отже, для найбільшого розміру алфавіту класів розпізнавання для одного визначення ArcFace буде потрібно 3,2/4000=0,0008с.=0,8мс, тобто 1/0,0008=1,250 ідентичностей можна передбачити вдруге за допомогою даної реалізації тестування. Якщо нам вдасться використати тестову реалізацію, настільки ж ефективну, то цей, час класифікації можна вважати мізерним

Таблиця 3.15-Розмір векторів функцій, сформований за допомогою 3 перевірених алгоритмів розпізнавання обличь.

Алгоритм	OpenFace	Dlib-R	ArcFace
Розмір векторної функції	128	128	512

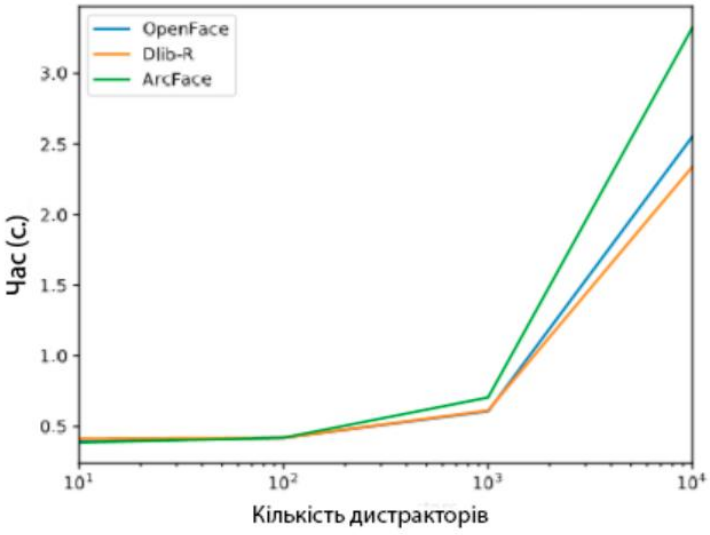


Рисунок 3.14 - Зростання часу класифікації (виражених в секундах) з кількістю дистракторів для трьох перевірених алгоритмів розпізнавання обличчя.

ВИСНОСКИ

В роботі розв’язано задачу інформаційного аналізу і синтезу системи інтелектуального аналізу відеоданих з метою виявлення і ідентифікації особи. При цьому виконано такі завдання:

1) Проаналізовано сучасний стан та тенденції розвитку технологій виявлення і індетифікації особи за зображенням.
2) Сформувано вхідного математичного опису системи інтелектуального аналізу відеоданих
3) Для детектування обличчя особи обрано 5 методів – метод Віоли-Джонса, метод HOG та три нейромережевих методи FRCNN, SFD, SSH.
4) Для ідентифікації особи за виділеним зображенням обличчя обрано 3 методи – OpenFace, Dlib-R, ArcFace
5) Як критерії оцінки ефективності методів детектування та ідентифікації запропоновано використовувати точність відповідних класифікаторів та оперативність їх навчання.
6) В роботі використано програмну реалізацію алгоритмів детектування та ідентифікації на мові програмування Python та C++.
7) Перевірка працездатності системи та порівняння ефективності алгоритмів на задачі детектування базою тестових наборів WIDER FACE та ідентифікації особи за базою MegaFace .

СПИСОК ЛІТЕРАТУРИ

[1] The mplab genki database, genki-4k subset. 1. 12

[2] Face and gesture recognition working group: Fg-net aging database. 2000. 23, 25

[3] A. S. Abdallah, M. A. El-Nasr, and A. L. Abbott. A new color image database for benchmarking of automatic face detection and human skin segmentation techniques. In Proceedings of World Academy of Science, Engineering and Technology, volume 20, pages 353–357. Citeseer, 2007. 12

[4] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. IEEE transactions on pattern analysis and machine intelligence, 28(12):2037–2041, 2006. 6

[5] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. CMU School of Computer Science, 2016. 8, 45

[6] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). Computer vision and image understanding, 110(3):346–359, 2008. 6

[7] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on pattern analysis and machine intelligence, 19(7):711–720, 1997. 8

[8] T. Berg and P. Belhumeur. Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 955–962, 2013. 33

[9] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers and identity-preserving alignment for face verification. In BMVC, volume 2, page 7. Citeseer, 2012. 33

[10] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y.-W. Teh, E. Learned-Miller, and D. A. Forsyth. Names and faces in the news. In Computer

Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–II. IEEE. 12, 20

[11] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In European Conference on Computer Vision, pages 354–370. Springer, 2016. 31

[12] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. arXiv preprint arXiv:1710.08092, 2017. 75

[13] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun. A practical transfer learning algorithm for face verification. In Computer Vision (ICCV), 2013 IEEE International Conference on, pages 3208–3215. IEEE, 2013. 33

[14] B.-C. Chen, C.-S. Chen, and W. H. Hsu. Cross-age reference coding for age-invariant face recognition and retrieval. In European Conference on Computer Vision, pages 768–783. Springer, 2014. 23

[15] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 3025–3032. IEEE, 2013. 33

[16] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In European Conference on Computer Vision, pages 109–122. Springer, 2014. 30

[17] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In Advances in Neural Information Processing Systems, pages 4470–4478, 2017. 47

[18] A. Colombo, C. Cusano, and R. Schettini. 3d face detection using curvature analysis. Pattern Recognition, 39(3):444 – 455, 2006. 6

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005. 6, 32, 37

[20] K. Davis. High quality face recognition with deep metric learning. 2017. i, 33

[21] J. Deng, J. Guo, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. arXiv preprint arXiv:1801.07698, 2018. i, 33, 46

[22] J. Deng, Y. Zhou, and S. Zafeiriou. Marginal loss for deep face recognition. In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPRW), Faces “in-the-wild” Workshop/Challenge, volume 4, 2017. 47

[23] C. Ding and D. Tao. Robust face recognition via multimodal deep face representation. IEEE Transactions on Multimedia, 17(11):2049–2058, 2015. 33

[24] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. Int J Comput Vis, 88:303–338, 2010. 3, 4, 5, 14, 15, 16

[25] S. S. Farfade, M. J. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, pages 643–650. ACM, 2015. 30

[26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence, 32(9):1627–1645, 2010. 6, 39

[27] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1):119–139, 1997. 35

[28] R. Girshick. Fast r-cnn. arXiv preprint arXiv:1504.08083, 2015. 39, 40

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014. 6, 38, 39, 41

[30] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. Image and Vision Computing, 28(5):807–813, 2010. 9

[31] P. J. Grother and M. L. Ngan. Face recognition vendor test (frvt) performance of face identification algorithms nist ir 8009. Technical report, 2014. 7, 25

[32] C. Gu, J. J. Lim, P. Arbela´ez, and J. Malik. Recognition using regions. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1030– 1037. IEEE, 2009. 38

[33] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A dataset and benchmark for large scale face recognition. In European Conference on Computer Vision. Springer, 2016. 75

[34] Z. Guo, Y.-N. Zhang, Y. Xia, Z.-G. Lin, Y.-Y. Fan, and D. D. Feng. Multi-pose 3d face recognition based on 2d sparse representation. Journal of Visual Communication and Image Representation, 24(2):117 – 126, 2013. Sparse Representations for Image and Video Analysis. 6, 8

[35] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In european conference on computer vision, pages 346– 361. Springer, 2014. 39, 40

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 46

[37] H. Hotelling. Analysis of a complex of statistical variables into principal components. Journal of educational psychology, 24(6):417, 1933. 8

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017. 47

[39] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507, 2017. 47

[40] P. Hu and D. Ramanan. Finding tiny faces. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1522–1530. IEEE, 2017. 31

[41] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, volume 1, page 3, 2017. 47

[42] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. 9, 19

[43] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. 3, 4, 5, 12, 13, 14

[44] H. Jiang and E. Learned-Miller. Face detection with the faster r-cnn. In Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on, pages 650–657. IEEE, 2017. 6

[45] T. Kanade. Picture processing system by computer complex and recognition of human faces. 1974. 8

[46] V. Kazemi and S. Josephine. One millisecond face alignment with an ensemble of regression trees. In 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, United States, 23 June 2014 through 28 June 2014, pages 1867–1874. IEEE Computer Society, 2014. 46, 72

[47] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4873–4882, 2016. i, 7, 24, 71

[48] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1931–1939, 2015. 17, 25

[49] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark

localization. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 2144–2151. IEEE, 2011. 3, 13

[50] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In Computer Vision, 2009 IEEE 12th International Conference on, pages 365–372. IEEE, 2009. 21

[51] G. P. Kusuma and C.-S. Chua. Pca-based image recombination for multimodal 2d+3d face recognition. Image and Vision Computing, 29(5):306 – 316, 2011. 6, 8

[52] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks, 8(1):98–113, 1997. 8

[53] G. B. H. E. Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014. 19

[54] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5325–5334, 2015. 6

[55] T.-Y. Lin, P. Doll´ar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In CVPR, volume 1, page 4, 2017. 43

[56] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016. 42

[57] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. arXiv preprint arXiv:1704.08063, 2017. 6, 33

[58] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In ICML, pages 507–516, 2016. 47

[59] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang. Recurrent scale approximation for object detection in cnn. In IEEE International Conference on Computer Vision, 2017. 29

[60] A. C. Loui, C. N. Judice, and S. Liu. An image database for benchmarking of automatic face detection and recognition algorithms. In Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on, volume 1, pages 146–150. IEEE, 1998. 12

[61] D. G. Lowe. Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on, volume 2, pages 1150–1157. Ieee, 1999. 38

[62] I. Masi, A. T. Tran, T. Hassner, J. T. Leksut, and G. Medioni. Do we really need to collect millions of faces for effective face recognition? In European Conference on Computer Vision, pages 579–596. Springer, 2016. 33

[63] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In European Conference on Computer Vision, pages 720–735. Springer, 2014. 24, 30

[64] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 922–928. IEEE, 2015. 8

[65] M. Najibi, P. Samangouei, R. Chellappa, and L. Davis. Ssh: Single stage headless face detector. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4875–4884, 2017. i, 31, 43, 44, 61

[66] M. Naphade, J. R. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. IEEE multimedia, 13(3):86–91, 2006. 16

[67] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In Image Processing (ICIP), 2014 IEEE International Conference on, pages 343–347. IEEE, 2014. 9, 25

[68] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In BMVC, volume 1, page 6, 2015. 46

[69] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 1(2):4, 2017. 8

[70] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, realtime object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016. 42

[71] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015. 38, 40, 41

[72] K. Ricanek and T. Tesafaye. Morph: A longitudinal image database of normal adult age-progression. In Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on, pages 341–345. IEEE, 2006. 23

[73] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 1, pages 746–751. IEEE, 2000. 12

[74] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 815–823, 2015. 8, 33, 45, 47

[75] M. P. Segundo, L. Silva, O. Bellon, and S. Sarkar. Orthogonal projection images for 3d face detection. Pattern Recognition Letters, 50:72 – 81, 2014. Depth Image Analysis. 6

[76] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs. Frontal to profile face verification in the wild. In Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on, pages 1–9. IEEE, 2016. 22

[77] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013. 42

[78] P. Sharma and R. B. Reilly. A colour face image database for benchmarking of automatic face detection algorithms. In Video/Image Processing and Multimedia Communications, 2003. 4th EURASIP Conference focused on, volume 1, pages 423–428. IEEE, 2003. 12

[79] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 57, 59, 62

[80] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. Josa a, 4(3):519–524, 1987. 8

[81] T.-H. Sun and F.-C. Tien. Using backpropagation neural network for face recognition with 2d+3d hybrid information. Expert Systems with Applications, 35(1):361 – 372, 2008. 8

[82] X. Sun, P. Wu, and S. C. Hoi. Face detection using deep learning: An improved faster rcnn approach. arXiv preprint arXiv:1701.08289, 2017. 29

[83] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1891–1898, 2014. 33

[84] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1891–1898, 2014. 47

[85] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In AAAI, volume 4, page 12, 2017. 47

[86] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015. 44, 45

[87] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to humanlevel performance in face verification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1701–1708, 2014. 10, 33, 45

[88] X. Tang, D. K. Du, Z. He, and J. Liu. Pyramidbox: A context-assisted single shot face detector. arXiv preprint arXiv:1803.07737, 2018. 29, 31

[89] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. 24

[90] M. Turk and A. Pentland. Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1):71–86, 1991. 8

[91] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. IEE Proceedings-Vision, Image and Signal Processing, 141(4):245–250, 1994. 6

[92] F. Van Lishout, A. Dubois, M. L. Wang, T. Ewbank, and L. Wehenkel. Integrating facial detection and recognition algorithms into real-life applications. Montefiore Institute - department of EE & CS, University of Li`ege, Belgium, 2018. Workshop on the Architecture of Smart Cameras - WASC 2018. 2

[93] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I–I. IEEE, 2001. 35

[94] P. Viola and M. J. Jones. Robust real-time face detection. International journal of computer vision, 57(2):137–154, 2004. 2, 6, 17, 20, 22, 25, 30, 32, 35, 36, 37

[95] S. Wan, Z. Chen, T. Zhang, B. Zhang, and K.-k. Wong. Bootstrapping face detection with hard negative examples. arXiv preprint arXiv:1608.02236, 2016. 29

[96] F. Wang, W. Liu, H. Liu, and J. Cheng. Additive margin softmax for face verification. arXiv preprint arXiv:1801.05599, 2018. 47

[97] H. Wang, Z. Li, X. Ji, and Y. Wang. Face r-cnn. arXiv preprint arXiv:1706.01061, 2017. 38

[98] J. Wang, Y. Yuan, and G. Yu. Face attention network: An effective face detector for the occluded faces. arXiv preprint arXiv:1711.07246, 2017. 31

[99] Y. Wang, X. Ji, Z. Zhou, H. Wang, and Z. Li. Detecting faces using region-based fully convolutional networks. arXiv preprint arXiv:1709.05256, 2017. 31

[100] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In European Conference on Computer Vision, pages 499–515. Springer, 2016. 47

[101] C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. Adams, T. Miller, N. Kalka, A. K. Jain, J. A. Duncan, K. Allen, et al. Iarpa janus benchmark-b face dataset. In CVPR Workshop on Biometrics, 2017. 17, 18, 25

[102] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 529–534. IEEE, 2011. 21

[103] W.-S. T. WST. Deeply learned face representations are sparse, selective, and robust. perception, 31:411–438, 2008. 45

[104] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1912–1920, 2015. 8

[105] Y. Xiong, K. Zhu, D. Lin, and X. Tang. Recognize complex events from static images by fusing deep channels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1600–1609, 2015. 16

[106] J. Yan, X. Zhang, Z. Lei, and S. Z. Li. Face detection by structural models. Image and Vision Computing, 32(10):790–799, 2014. 3, 14

[107] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Aggregate channel features for multi-view face detection. In Biometrics (IJCB), 2014 IEEE International Joint Conference on, pages 1–8. IEEE, 2014. 30

[108] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Fine-grained evaluation on face detection in the wild. In Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on, volume 1, pages 1–7. IEEE, 2015. 3, 5, 15, 16

[109] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. IEEE Transactions on pattern analysis and machine intelligence, 24(1):34–58, 2002. 3

[110] S. Yang, P. Luo, C.-C. Loy, and X. Tang. From facial parts responses to face detection: A deep learning approach. In Proceedings of the IEEE International Conference on Computer Vision, pages 3676–3684, 2015. 30

[111] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wider face: A face detection benchmark. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5525–5533, 2016. i, 3, 16

[112] S. Yang, Y. Xiong, C. C. Loy, and X. Tang. Face detection through scale-friendly deep convolutional networks. arXiv preprint arXiv:1706.02863, 2017. 29

[113] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. arXiv preprint arXiv:1411.7923, 2014. 46

[114] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In European conference on computer vision, pages 818–833. Springer, 2014. 45

[115] K. Zhang, Z. Zhang, H. Wang, Z. Li, Y. Qiao, and W. Liu. Detecting faces using inside cascaded contextual cnn. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3171–3179, 2017. 29

[116] N. Zhang, M. Paluri, Y. Taigman, R. Fergus, and L. Bourdev. Beyond frontal faces: Improving person recognition using multiple cues. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4804–4813, 2015. 23 [117] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S^3FD:

Single shot scale-invariant face detector. CoRR, abs/1708.05237, 2017. 29, 31, 42, 43

[118] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao. Range loss for deep face recognition with long-tail. arXiv preprint arXiv:1611.08976, 2016. 47

[119] C. Zhu, R. Tao, K. Luu, and M. Savvides. Seeing small faces from robust anchor's perspective. arXiv preprint arXiv:1802.09058, 2018. 31

[120] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2879–2886. IEEE, 2012. 3, 14

[121] C. L. Zitnick and P. Dolla´r. Edge boxes: Locating object proposals from edges. In European Conference on Computer Vision, pages 391–405. Springer, 2014. 16

ДОДАТОК

```
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
using Windows.Graphics.Imaging;
using Windows.Media.FaceAnalysis;
using Windows.UI.Xaml.Media.Imaging;
using Windows.UI.Xaml.Shapes;

FaceDetector faceDetector;
IList<DetectedFace> detectedFaces;
FileOpenPicker photoPicker = new FileOpenPicker();
photoPicker.ViewMode = PickerViewMode.Thumbnail;
photoPicker.SuggestedStartLocation = PickerLocationId.
PicturesLibrary;
photoPicker.FileTypeFilter.Add(".jpg");
photoPicker.FileTypeFilter.Add(".jpeg");
photoPicker.FileTypeFilter.Add(".png");
photoPicker.FileTypeFilter.Add(".bmp");

StorageFile photoFile = await photoPicker.PickSingleFileAsync
();
if (photoFile == null)
{ return;}

IRandomAccessStream fileStream = await photoFile.OpenAsync(
FileAccessMode.Read);
BitmapDecoder decoder = await BitmapDecoder.CreateAsync(
fileStream);

BitmapTransform transform = new BitmapTransform();
const float sourceImageHeightLimit = 1280;
if (decoder.PixelHeight > sourceImageHeightLimit)
{
  float scalingFactor = (float)sourceImageHeightLimit / (
  float)decoder.PixelHeight;
  transform.ScaledWidth = (uint)Math.Floor(decoder.PixelWidth
   * scalingFactor);
  transform.ScaledHeight = (uint)Math.Floor(decoder.
  PixelHeight * scalingFactor);
}
SoftwareBitmap sourceBitmap = await decoder.
GetSoftwareBitmapAsync(decoder.BitmapPixelFormat,
```

```
BitmapAlphaMode.Premultiplied, transform, ExifOrientationMode
.IgnoreExifOrientation, ColorManagementMode.DoNotColorManage)

// Use FaceDetector.GetSupportedBitmapPixelFormats and
IsBitmapPixelFormatSupported to dynamically
// determine supported formats
const BitmapPixelFormat faceDetectionPixelFormat =
BitmapPixelFormat.Gray8;

SoftwareBitmap convertedBitmap;

if (sourceBitmap.BitmapPixelFormat !=
faceDetectionPixelFormat)
{
  convertedBitmap = SoftwareBitmap.Convert(sourceBitmap,
  faceDetectionPixelFormat);
}
else
{
  convertedBitmap = sourceBitmap;
}
if (faceDetector == null)
{
  faceDetector = await FaceDetector.CreateAsync();
}

detectedFaces = await faceDetector.DetectFacesAsync(
convertedBitmap);
ShowDetectedFaces(sourceBitmap, detectedFaces);

Видалимо об'єкти, створені під час процесу виявлення осіб.

sourceBitmap.Dispose();
fileStream.Dispose();
convertedBitmap.Dispose();


<Canvas x:Name="VisualizationCanvas" Visibility="Visible"
Grid.Row="0" HorizontalAlignment="Stretch" VerticalAlignment=
"Stretch"/>

// Визначимо кілька змінних-членів, щоб оформити квадрати,
```

```
які будуть намальовані.
private readonly SolidColorBrush lineBrush = new
SolidColorBrush(Windows.UI.Colors.Yellow);
private readonly double lineThickness = 2.0;
private readonly SolidColorBrush fillBrush = new
SolidColorBrush(Windows.UI.Colors.Transparent);

private async void ShowDetectedFaces(SoftwareBitmap
sourceBitmap, IList<DetectedFace> faces)
{
  ImageBrush brush = new ImageBrush();
  SoftwareBitmapSource bitmapSource = new
  SoftwareBitmapSource();
  await bitmapSource.SetBitmapAsync(sourceBitmap);
  brush.ImageSource = bitmapSource;
  brush.Stretch = Stretch.Fill;
  this.VisualizationCanvas.Background = brush;

  if (detectedFaces != null)
  {
    double widthScale = sourceBitmap.PixelWidth / this.
    VisualizationCanvas.ActualWidth;
    double heightScale = sourceBitmap.PixelHeight / this.
    VisualizationCanvas.ActualHeight;

    foreach (DetectedFace face in detectedFaces)
    {
      // Create a rectangle element for displaying the face
      box but since we're using a Canvas
      // we must scale the rectangles according to the
      image's actual size.
      // The original FaceBox values are saved in the
      Rectangle's Tag field so we can update the
      // boxes when the Canvas is resized.
      Rectangle box = new Rectangle();
      box.Tag = face.FaceBox;
      box.Width = (uint)(face.FaceBox.Width / widthScale);
      box.Height = (uint)(face.FaceBox.Height / heightScale);
      box.Fill = this.fillBrush;
      box.Stroke = this.lineBrush;
      box.StrokeThickness = this.lineThickness;
      box.Margin = new Thickness((uint)(face.FaceBox.X /
```

```
        widthScale), (uint)(face.FaceBox.Y / heightScale), 0, 0
        );

        this.VisualizationCanvas.Children.Add(box);
      }
    }
  }

using Windows.Media;
using System.Threading;
using Windows.System.Threading;
private FaceTracker faceTracker;
private ThreadPoolTimer frameProcessingTimer;
private SemaphoreSlim frameProcessingSemaphore = new
SemaphoreSlim(1);

this.faceTracker = await FaceTracker.CreateAsync();
TimeSpan timerInterval = TimeSpan.FromMilliseconds(66); //
15 fps
this.frameProcessingTimer = Windows.System.Threading.
ThreadPoolTimer.CreatePeriodicTimer(new Windows.System.
Threading.TimerElapsedHandler(ProcessCurrentVideoFrame),
timerInterval);
public async void ProcessCurrentVideoFrame(ThreadPoolTimer
timer)
{
  if (!frameProcessingSemaphore.Wait(0))
  {
    return;
  }
  VideoFrame currentFrame = await GetLatestFrame();
  // Use FaceDetector.GetSupportedBitmapPixelFormats and
  IsBitmapPixelFormatSupported to dynamically
  // determine supported formats
  const BitmapPixelFormat faceDetectionPixelFormat =
  BitmapPixelFormat.Nv12;

  if (currentFrame.SoftwareBitmap.BitmapPixelFormat !=
  faceDetectionPixelFormat)
  {
    return;
  }
```

```
try
{
  IList<DetectedFace> detectedFaces = await faceTracker.
  ProcessNextFrameAsync(currentFrame);

  var previewFrameSize = new Windows.Foundation.Size(
  currentFrame.SoftwareBitmap.PixelWidth, currentFrame.
  SoftwareBitmap.PixelHeight);
  var ignored = this.Dispatcher.RunAsync(Windows.UI.Core.
  CoreDispatcherPriority.Normal, () =>
  {
    this.SetupVisualization(previewFrameSize, detectedFaces
    );
  });
}
catch (Exception e)
{
  // Face tracking failed
}
finally
{
  frameProcessingSemaphore.Release();
}

currentFrame.Dispose();
```