Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж»

Виконав: студент 2 курсу,
групи 1КН-18 м
спеціальності 122 «Комп'ютерні науки»
Мазур М.В.
Керівник: к.т.н., доц. Колесницький О. К.
Рецензент: к.т.н., доц. Рейда О. М.

Вінниця - 2019 року

ЗАТВЕРДЖУЮ
Директор ТОВ «ІНКОРСОФТ»
Сивинюк О. В.
(наук. ст., вч. зв., ініц. та прізв.) (підпис)
"____"_____2019 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д. т. н., проф. Яровий А.А.
(наук. ст., вч. зв., ініц. та прізв.) (підпис)
"____"_____2019 р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.009.18.000.ПЗ

Магістранта групи 1КН-18м Мазура Максима Віталійовича

Тема магістерської кваліфікаційної роботи: «Детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж»

Вхідні дані: роздільна здатність зображень не більше 2048×1536 пікселів, кількість шарів мережі не менше 10, формат вхідного зображення png, jpg, bmp, час обробки не більше 50 мс, мова програмування об'єктно-орієнтована, програмна бібліотека комп'ютерного зору OpenCV, програмно-апаратна платформа NVIDIA CUDA.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: схема загального алгоритму роботи програми детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж, структура програми детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж, діаграма класів програми детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж; структура згорткової нейронної мережі, результати навчання згорткової нейронної мережі детектування облич, фрагмент коду програми розпізнавання облич на зображеннях на основі згорткових нейронних мереж, приклади роботи програми.

2. Текстова (пояснювальна записка): вступ, аналіз предметної області детектування облич на зображеннях, розробка процесів детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж, проектування системи детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж, економічна частина, висновки, перелік використаних джерел, додатки.

АНОТАЦІЯ

В даній магістерській кваліфікаційній роботі було реалізовано програму детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж.

Було проведено огляд сучасних методів детектування та відомих програмних реалізацій розпізнавання облич. Було розроблено процеси необхідні для розпізнавання облич, а саме архітектуру згорткової мережі та математичну модель відсікання надлишкових фільтрів в мережі. Результатом проектування є програма. Програмний додаток реалізовано мовою Python в середовищі PyCharm.

ABSTRACT

In this master's qualification, a program for detecting and recognizing faces on images based on convolutional neural networks was implemented.

Modern methods of detection and known software implementations of face recognition were conducted. The processes necessary for face recognition were developed, namely the convolution network architecture and the mathematical model for cutting off excess filters in the network. The result of the design is the program. The application is implemented in Python in PyCharm.

ЗМІСТ

ВСТУП

Актуальність дослідження. Поняття штучного інтелекту зародилося багато десятків років тому. В свідомості людей воно осідало під впливом наукової і художньої літератури, кіно тощо. Люди мріяли створити штучний розум, який вирішує будь-яку задачу, але на даний момент людство на етапі, який знаходиться доволі далеко, від омріяної точки наукового неповернення, з якого штучний інтелект набуде самостійної свідомості, і буде рухати науковий прогрес самостійно. Проте, сучасний світ вже важко уявити без штучного інтелекту. Це поняття тісно переплелося в буденних реаліях майже кожної людини, хоча не всі здогадуються про це. Ця наукова область рекордсмен по темпам розвитку. Наукові відкриття в ній робляться майже кожного дня. Являючись порівняно молодою сферою з широким спектром наукових проблем, вона містить багато потенційних відкриттів, які будуть з'являтися найближчим часом, змінюючи взаємодію людини з навколишнім світом.

Особливо актуальними стають задачі, що мають некоректну постановку, для яких відсутні оптимальні алгоритми їх розв'язку. Власне кажучи, при розв'язуванні саме таких задач найчастіше використовують штучні нейронні мережі (ШНМ) — один із методів реалізації штучного інтелекту.

Сучасні штучні нейронні мережі складаються з великої кількості простих процесорних елементів із деякою кількістю локальної пам'яті (нейронів), об'єднаних за допомогою дискретних або неперервних комунікаційних каналів. Задачі, що розв'язують на ШНМ, підлягають декомпозиції на множину локальних задач, кожна з яких може бути розв'язаною за допомогою окремого нейрона шляхом реалізації певного алгоритму обробки локальних даних.

Сьогодні нейронні мережі вирішують різні задачі, але є одна основна — допомога людині в її діяльності. Одною з найбільш пріоритетних задач в кожному місті є безпека його жителів. В кожному великому місті уже роблять свої різноманітні програми „безпечного міста". Найбільш поширеним рішенням є встановлення камер відеоспостереження в громадських місцях.

Таким чином можна дізнатися, що відбувалося в момент різних правопорушень або дізнатись хто їх здійснив. У великих містах встановлено десятки (в деяких навіть сотні) тисяч камер і їх моніторинг в реальному часі є неможливим. В таких випадках на допомогу приходять рішення із сфери комп'ютерного зору. Таким чином можна в масштабувати кількість камер до необхідного рівня, без збільшення людських ресурсів, а також робити це в реальному часі, в момент або до моменту скоєння правопорушення. Ці переваги автоматизації роблять задачі комп'ютерного зору найбільш актуальними задачами у сфері „безпечного міста".

Однією із задач, вирішення яких робить місто безпечнішим є розпізнавання людей на камерах відеоспостереження, які знаходяться у розшуку з різних причин. Дана задача також може бути успішно задіяна в маркетингових цілях.

Для вирішення задачі ідентифікації людини потрібно вирішити задачі детектування обличчя та класифікації особи. Існує вже досить багато реалізацій даних задач з достатньо точним результатом. Основна проблема яку намагаються вирішити дослідники в даній сфері, це зменшення кількості обчислювальних потужностей без значної втрати точності.

Більшість обчислювальних ресурсів комп'ютера необхідні для вирішення задачі детектування обличчя. Гостро стоїть задача зробити час детектування якомога меншим для того, щоб збільшити кількість відеопотоків, які обробляються одночасно. Таким чином оптимізація вже готових методів детектування є актуальною.

Найбільш популярними і широко використовуваними системами детектування є системи на базі згорткових нейронних мереж. Згорткові мережі використовуються через їхню точність і гнучкість, при цьому обчислювальні потужності які вони використовують є допустимими на сьогоднішній день. Обчислювальна точність не завжди прямо залежить від кількості обчислювальних операцій.

Зв'язок роботи з науковими програмами, планами, темами.

Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення швидкодії детектування та розпізнавання облич на зображеннях.

Для досягнення наведеної мети треба вирішити наступні задачі:

1) Дослідити процес розпізнавання обличчя та оглянути основні програми розпізнавання;
2) Оглянути основні методи детектування, та проаналізувати існуючі варіанти для детектування;
3) Розробити архітектуру згорткової нейронної мережі для детектування облич та математичну модель методу для зменшення кількості обчислень;
4) Розробити метод детектування облич на зображеннях;
5) Розробити структуру та алгоритм роботи програмного засобу для детектування та розпізнавання облич на зображеннях;
6) Обґрунтувати вибір інструментів розробки;
7) Розробити додаток який демонструє результати проведених робіт.
8) Провести тестування розробленої програми та проаналізувати результати тестування.

Об'єкт дослідження – процес детектування та розпізнавання облич на зображеннях.

Предмет дослідження – це методи та програмні засоби детектування та розпізнавання облич на зображеннях та їх швидкодія.

Методи дослідження. У роботі використані наступні методи наукових досліджень: теорія штучних нейронних мереж, теорія розпізнавання образів, теорія моделювання систем, теорія машинного навчання, теорія паралельних

обчислень, теорія векторної кластеризації.

Наукова новизна одержаних результатів полягає в наступному:

Вдосконалено структуру згорткової нейронної мережі за рахунок відсікання фільтрів, які несуттєво впливають на результат детектування.

Змінено кількість обчислень шляхом зменшення кількості якорів, для вибору зони розташування об’єктів.

Ці покращення дозволили підвищити швидкодію детектування та розпізнавання облич на зображеннях.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено архітектуру мережі для детектування облич на зображеннях;
2. Розроблено алгоритм розпізнавання облич на зображеннях на основі згорткових нейронних мереж;
3. Розроблено програмний продукт для детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж.

Розроблений програмний засіб був протестований та плануються до впровадження на ТОВ «ІНКОРСОФТ УКРАЇНА», про що є відповідна довідка – див. додаток Г.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, порівнянням результатів з результатами програм аналогів, збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, здобуті самостійно. У роботах, опублікованих у співавторстві, магістранту належать такі результати: [1] – результати застосування нейронних мереж в сучасних додатках; [2] – експериментальні дослідження використання різних типів нейронних мереж для вирішення задачі розпізнавання образів; [3] – розробка покращеної

функції активації нейронної мережі для вирішення задачі класифікації образів; [4] – програмна реалізації системи моделювання нейронного елементу для класифікації динамічних образів.

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2019)», XLVIII науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії у березні 2019 м. Вінниці, XLVII науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії у березні 2018 м. Вінниці.

Публікації. За результатами досліджень отримано свідоцтво про реєстрацію авторського права на твір – комп'ютерна програма «Моделювання спайкінгового нейрона» [4], опубліковано троє тез доповідей науково-технічних конференцій [1-3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДЕТЕКТУВАННЯ ТА РОЗПІЗНАВАННЯ ОБЛИЧ НА ЗОБРАЖЕННЯХ

1.1 Процес розпізнавання облич

Технології розпізнавання обличчя почали розроблятися ще в далекому 1964 році. В той час Хелен Чан і Чарльз Біссон працювали на невелику компанію і мали на меті розпізнавати обличчя людей за допомогою комп'ютера. Опираючись на документацію, яка описувала їхню роботу, принцип полягав у довільному розміченні ключових точок на обличчі, таких як очі і губи та математичному розвертанні обличчя, щоб компенсувати повороти обличчя. Відстань між цими точками теж математично обраховувалась і зберігалась. Проблема полягала у виборі із великої бази зображень тих, чий набір даних про обличчя збігався з шуканим екземпляром. Успіх системи був зав'язаний на кількості облич в базі для пошуку. Систему називали на пів комп'ютерною, на пів людською, адже людина помічала координати основних точок на зображеннях, а згодом комп'ютер порівнював ці обличчя, і повертав найбільш близькі результати. Також система мала недоліки, через те, що умови в який були зроблені зображення були не ідеальними. Голова людини не завжди була повернута фронтально і мала довільне положення в просторі. Через це повинно було задаватись координати точок голови в просторі, для більш точного перетворення координатів [5].

Усі системи розпізнавання облич в реальному часі мають на меті, отримати із вхідного потоку зображень результат в формі ідентифікатора особи. Цей процес включає в себе два основних етапи — детектування та розпізнавання.

Детектування об'єктів — це задача сфери комп'ютерних технологій, що відноситься до класу задач комп'ютерного зору і обробки зображень. Ця задача вирішує проблему пошуку фрагментів семантичних об'єктів, що відносяться до конкретної множини. Кожен об'єкт має набір власних

візуальних характеристик, таких як: кольори, форми та взаєморозміщення відносно інших об’єктів з такими ж характеристиками. Для прикладу можна представити об’єкт дерево, який складається зі стовбура, який є зазвичай прямої форми, має коричневий колір і знаходиться над травою, а над стовбуром розміщене зелене листя, невеликого розміру, яке розкидане хаотично. Таким чином у нас є набір візуальних характеристик дерева і ми знаємо які властивості слід виділити при детектуванні дерев.

Віднесення характеристик зображення до певного класу називається класифікацією, але окрім класифікації при проведенні детекції слід виділити межі об’єкту на зображеннях, це дозволяє детектувати більше одного об’єкта на зображенні [6].

В нашому випадку, для проведення розпізнавання не достатньо лише класифікації об’єкта. Для того, щоб відокремити одну людину від іншої, ми маємо заздалегідь знати набір візуальних характеристик особи і до кого вони відносяться. В даному випадку нам потрібно виділяти конкретні ознаки і порівнювати їх між собою, щоб спевнитись співпадають вони, чи ні. Цей процес називається розпізнаванням [7].

Окрім детектування і розпізнавання застосовуються й інші методи, які покращують кінцевий результат. До цих методів відноситься: сегментація та відокремлення об’єкту від фону, накладання різних фільтрів, вирівнювання, афінні перетворення, зміна роздільної здатності об’єктів, для детектування на різних рівняй деталізації та інші.

Основними критеріями якими керуються під час вибору методів і алгоритмів детектування і розпізнавання є точність, швидкодія, універсальність і стійкість до перешкод.

Тематикою даної кваліфікаційної роботи буде підвищення одного з цих показників, а саме швидкодії детектування.

1.2 Огляд основних методів детектування облич

Незважаючи на значні останні досягнення в області розпізнавання обличчя, що здійснює перевірку особи, ефективне розпізнавання в великих масштабах є великою проблемою і дає виклик сучасним підходам. Спеціалісти в сфері комп’ютерного зору стикаються з визначенням пріоритетів при розпізнаванні. Точність алгоритмів розпізнавання дуже сильно пов’язане з кількістю обчислювальних операцій необхідних для отримання результатів.

Алгоритм Viola-Jones є широко використовуваним механізмом для виявлення об'єктів. Головною властивістю цього алгоритму є те, що навчання відбувається повільно, але виявлення відбувається дуже швидко. Цей алгоритм використовує фільтри функції Хаара, в зв’хку з цим операція множення не використовується. Таким чином економиться значна частина обчислювальних ресурсів. На рисунку 1.3 зображено процес детектування [8].

Ефективність алгоритму Віола-Джонса може бути значно збільшена, якщо спочатку згенерувати інтегральне зображення. Даний алгоритм використовується в навчальних цілях, або для здійснення детектування або виділення особливостей нескладних задач комп’ютерного зору.

$$II(y,x) = \sum_{p=0}^{y} \sum_{q=0}^{x} Y(p,q) \tag{1.1}$$

$$C_m = \begin{cases} 1, & \sum_{i=0}^{I_m-1} F_{m,i} > \theta_m \\ 0, & \text{otherwise} \end{cases}$$

$$F_{m,i} = \begin{cases} \alpha_{m,i}, & \text{if } f_{m,i} > t_{m,i} \\ \beta_{m,i}, & \text{otherwise} \end{cases} \tag{1.2}$$

Інтегральні зображення для екстракторів Хаара розраховуються шляхом додавання лише чотирьох чисел. Наприклад, інтеграл зображення області ABCD (рисунок1) обчислюється як f(yA, xA) - f(yB, xB) - f(yC, xC) + f(yD, xD).

Даний метод виділяє області з характерним розташуванням пікселів на зображенні і шляхом складних обчислень класифікує об’єднані зони до певного класу зображень (рисунок 1.1, 1.2)(формула 1.1, 1.2).

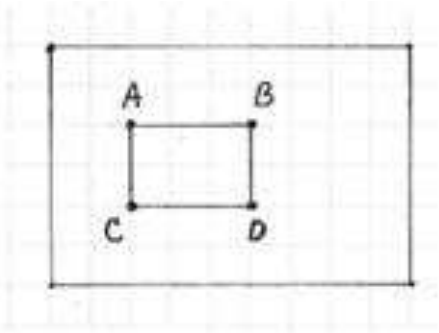

Рисунок 1.1 — Інтегрування області зображення з використанням інтегрованого зображення

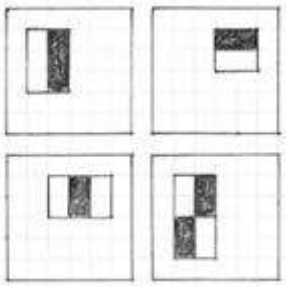

Рисунок 1.2 — Приклад прямокутних областей рис на зображенні

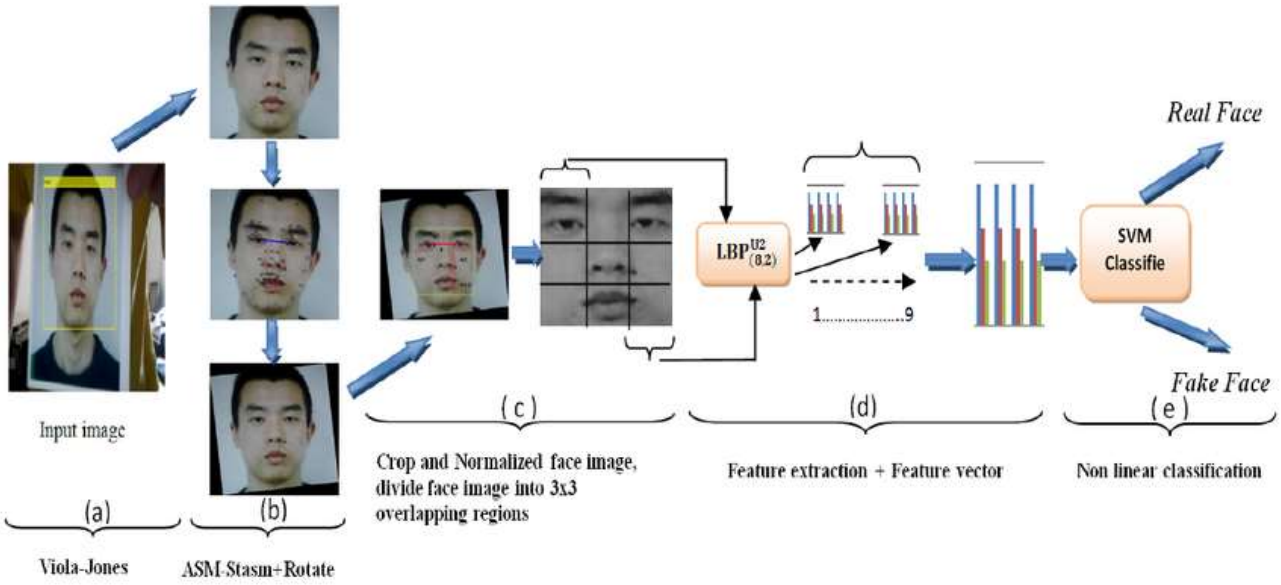

Рисунок 1.3 — Процес розпізнавання за допомогою метода Віоли-Джонса

Набагато ефективніша методика виявлення (гістограми орієнтованих градієнтів, 2005). Navneet Dalal і Bill Triggs винайшли "HOG" для виявлення пішоходів. Їх дескриптор ознак, Гістограми орієнтованих градієнтів (HOG), значно перевершив існуючі алгоритми в цьому завданні [9].

Для кожного окремого пікселя алгоритм дивиться на пікселі, які безпосередньо оточують його (рисунок 1.4).

Мета полягає в тому, щоб дізнатись наскільки темним є поточний піксель у порівнянні з оточуючими пікселями. Ми повторюємо цей процес для кожного окремого пікселя зображення. Кожен піксель замінюється стрілкою. Ці стрілки називаються градієнтами. Градієнти показують потік від світлого до темного по всьому зображенню. Ми розбиваємо зображення на маленькі квадрати 16x16 пікселів кожен. На кожному квадраті ми будемо підраховувати кількість точок градієнтів у кожному з основних напрямків Потім ми замінимо цей квадрат на зображення стрілками, які були найтемнішими.

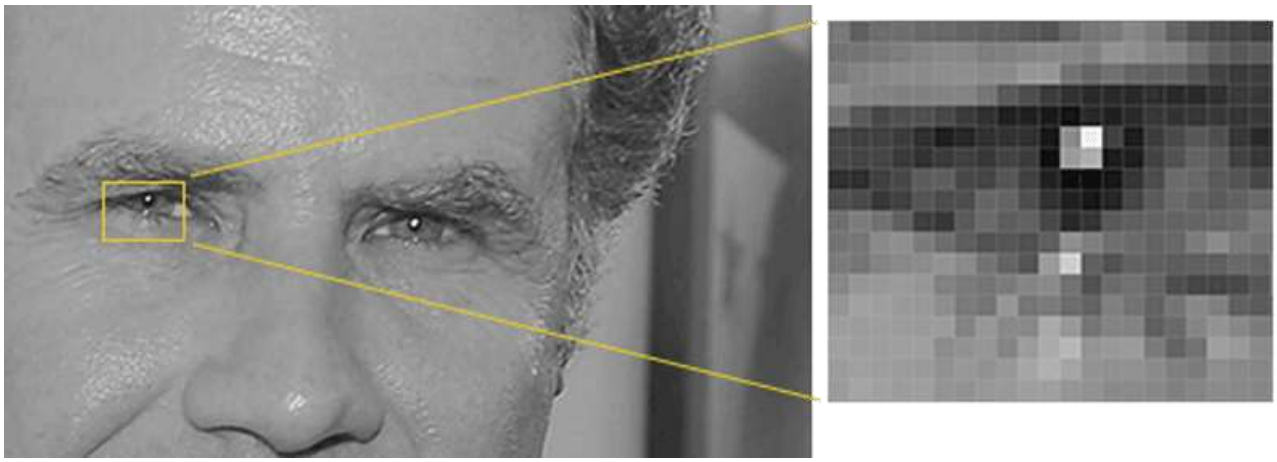

Рисунок 1.4 — Метод орієнтованих градієнтів

Кінцевий результат — оригінальне зображення, перетворене в просте представлення, яке простим чином фіксує основну структуру обличчя.

Виявлення облич означає знайти частину нашого зображення, яка виглядає найбільш схожою на відомий шаблон HOG, витягнутий з групи інших тренувальних осіб

У даній роботі, яка безпосередньо вивчає перетворення зображення обличчя у компактну модель евклідового простору, де відстані безпосередньо відповідають мірі схожості обличчя. Цей простір було створено для таких завдань, як розпізнавання облич. Перевірка і кластеризація, можуть бути легко реалізовані з використанням стандартних методів векторів ознак [10].

Найсучаснішими методами детектування сьогодні являються методи базовані на глибокому навчанні. Ці методи базуються на використанні архітектур згорткових нейронних мереж. Даний підхід є найбільш точним і задовольняє більшість потреб при виборі методу детектування, а також є можливим дял детектування об’єктів в реальному часі завдяки сучасним обчислювальним можливостям комп’ютера.

Згорткові нейронні мережі (ЗНМ, англ. convolutional neural network, CNN, ConvNet) в машинному навчанні — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень.

ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоби вимагати використання мінімального обсягу попередньої обробки. Вони відомі також як інваріантні відносно зсуву (англ. shift invariant) або просторово інваріантні штучні нейронні мережі, виходячи з їхньої архітектури спільних ваг та характеристик інваріантності відносно паралельного перенесення.

На створення згорткових мереж, вчених надихнули біологічні процеси, в яких схему з'єднання нейронів запозичено з організації зорової кори тварин. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле.

ЗНМ використовують порівняно мало попередньої обробки, в порівнянні з іншими алгоритмами класифікації зображень. Це означає, що мережа навчається завдяки побудові і навчанню фільтрів, що в традиційних алгоритмах детектування конструювали вручну, що робило систему детектування меш самостійною. Ця незалежність у конструюванні ознак від апріорних знань та людських зусиль є великою перевагою.

1.3 Аналіз найпопулярніших згорткових мереж для детектування облич

Багато архітектур згорткових нейронних мереж були розроблені ще десятки років тому. Єдине, чого не було у дослідників на той час це потужностей для виконання такої кількості обчислень необхідних для проходження зображень через мережу. Так було до 2012 року коли згорткова мережа AlexNet здійснила революцію в області розпізнавання зображень.

AlexNet — це ім’я згорткової нейронної мережі розробленої Олексієм Крижевським і опублікована разом з його наставником, який спочатку ставився скептично до ідеї свого учня.

30 вересня 2012 року мережа взяла участь у змаганні візуального розпізнавання ImageNet Large Scale. Мережа досягла помилки в топ-5 на 15,3%, що на 10,8 відсотків менше, ніж у другого місця змагання. Таким чином, стрибок точності над іншими мережами здійснив ажіотаж навколо даної мережі [11].

Основним результатом оригінальної статті було те, що глибина моделі була досить високою для її високої продуктивності, що було обчислювально дорого, але стало можливим завдяки використанню блоків графічної обробки (GPU) під час навчання. Дана робота стала найбільш відомою роботою по згортковим нейронним мережам і станом на 2019 рік налічує більше 47 тисяч переглядів. AlexNet стала початком для створення ще кращих і більш продуктивних мереж, а також привернула увагу дослідників, і цим самим визвавши великий в сфері технологій.

1.3.1 Faster R-CNN

Сучасні мережі виявлення об’єктів залежать від алгоритмів пропозиції регіону для висунення гіпотетичного розташування об’єктів. Такі досягнення, як SPPnet та Fast R-CNN, скоротили час роботи цих мереж виявлення, звузивши область обчислення пропозицій, що зробило її менш ресурсно затратною. Дана мережа впроваджує поняття регіональних пропозицій (RPN), яка поділяє повне зображення згортки з мережею виявлення, завдяки чому пропонуються майже незатратні пропозиції регіону. RPN – це повністю згорткова мережа, яка одночасно прогнозує межі об’єктів і об’єктивні оцінки в кожній позиції. RPN тренується від методом повної ітерації навчання всіх шарів, генерують високоякісні пропозиції регіонів, які використовуються Fast R-CNN для детектування. Далі об’єднуються RPN і Fast R-CNN в єдину мережу, обмінюючись своїми згортковими ознаками — використовуючи досить відому модель нейронних мереж VGG-16 [12].

Механізми «зони уваги», компонент RPN повідомляє єдиній мережі, де його шукати. Для дуже глибокої моделі VGG-16, система виявлення має

частоту детектування 5 кадрів в секунду (включаючи всі кроки) на графічному процесорі, при цьому досягаючи конкурентного рівня виявлення об’єктів з точністю наборів даних PASCAL VOC 2007, 2012 та MS COCO із лише 300 пропозиціями на зображення. В конкурсах ILSVRC та COCO 2015 року, Faster R-CNN та RPN — взяли перші місця в декількох номінаціях.

На рисунку 1.5 показано повну архітектуру Faster R-CNN. Як можна побачити спочатку йдуть основні згорткові шари VGG-16, а після її виходів йдуть об’єднані моделі RPN та Fast R-CNN.

Дана модель деякий час була найкраща у відношенні швидкості до точності. Але на сьогоднішній день існують мережі, які при невеликій втраті точності потребують менших обчислювальних потужностей.

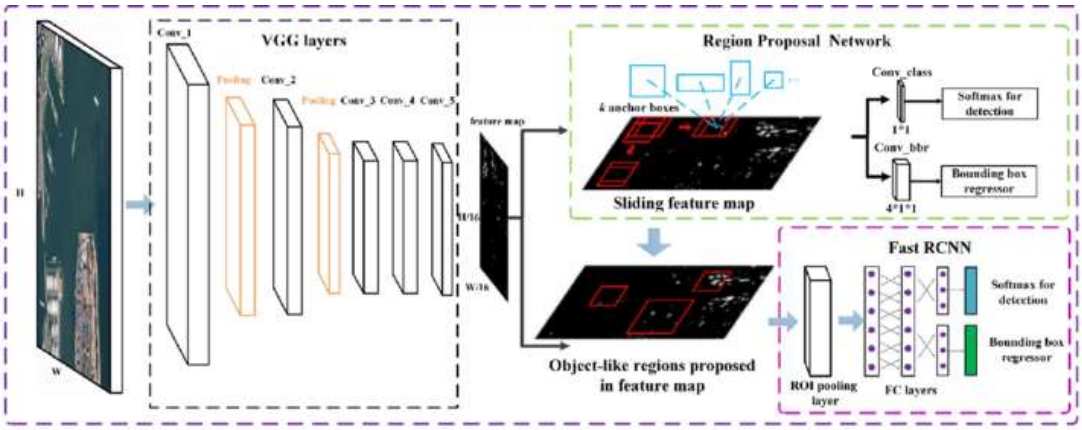

Рисунок 1.5 — Архітектура Faster R-CNN

В сучасних реаліях розробникам необхідно підтримувати швидкість розпізнавання в межах реального часу. Такі мережі використовують менше шарів в середині, але алгоритми цих шарів, компенсують їх кількість якістю.

1.3.2 SSD MobileNet

Нещодавно дослідники Google анонсували мережу MobileNet версії 2. Це, в основному, вдосконалення першої версії, що робить його ще більш ефективним та потужним. Хоча й перша версія показувала досить серйозні

результати в плані продуктивності , друга версія, ще краща зі значним відривом [13].

Основна ідея MobileNet V1 полягає в тому, що згорткові шари, які є важливими для завдань комп'ютерного зору, але досить дорогі для обчислення, можуть бути замінені так званими глибоко відокремлюваними згортками.

Завдання шару згортки розділено на дві підзадачі: спочатку є шар глибокої згортки, який фільтрує вхід, а потім шар згортання 1 × 1 (або в точці), який поєднує ці відфільтровані значення для створення нових функцій, як показано на рисунку 1.6.

Разом згортання по глибині і точкова згортка утворюють блок «згортання по глибині». Це робить приблизно те саме, що і традиційна згортка, але набагато швидше.

Повна архітектура MobileNet V1 складається з регулярної 3 × 3 згортки як першого шару, після чого в 13 разів більше зазначеного будівельного блоку.

Між цими блоками, що розділяються по глибині, немає шарів об'єднання. Натомість деякі з глибинних шарів мають крок 2, щоб зменшити просторові розміри даних. Коли це відбувається, відповідний точковий шар також подвоює кількість вихідних каналів. Якщо вхідне зображення становить 224 × 224 × 3, то вихід мережі — це карта 7 × 7 × 1024.

Як це часто зустрічається в сучасних архітектурах, за шарами згортки йде послідовна нормалізація. Функція активації, яка використовується MobileNet, є ReLU6. Це як у відомої ReLU, але це запобігає надмірній активації: y = min(max(0, x), 6).

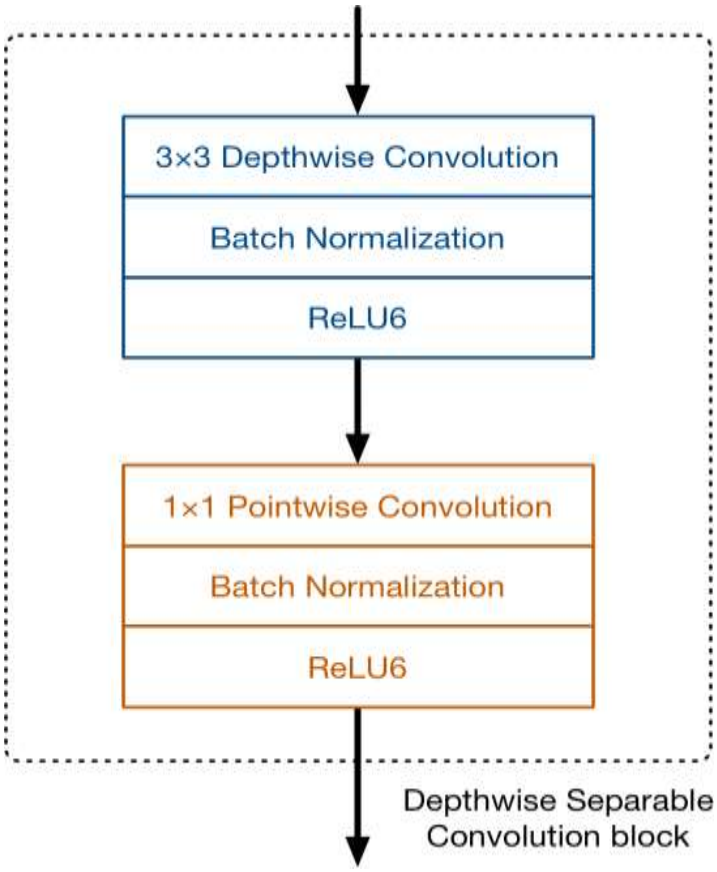

Рисунок 1.6 — Блок глибокої згортки

Автори статті MobileNet виявили, що ReLU6 є більш надійним, ніж звичайний ReLU при використанні обчислень з низькою точністю. Це також робить форму функції більш схожою на сигмоподібну: Функція активації ReLU6, зображена на рисунку 1.7.

У класифікаторі, заснованому на MobileNet, як правило, на самому кінці є середній загальний шар об’єднання, який супроводжується повністю пов’язаним класифікаційним шаром або еквівалентною згорткою 1 × 1 та softmax.

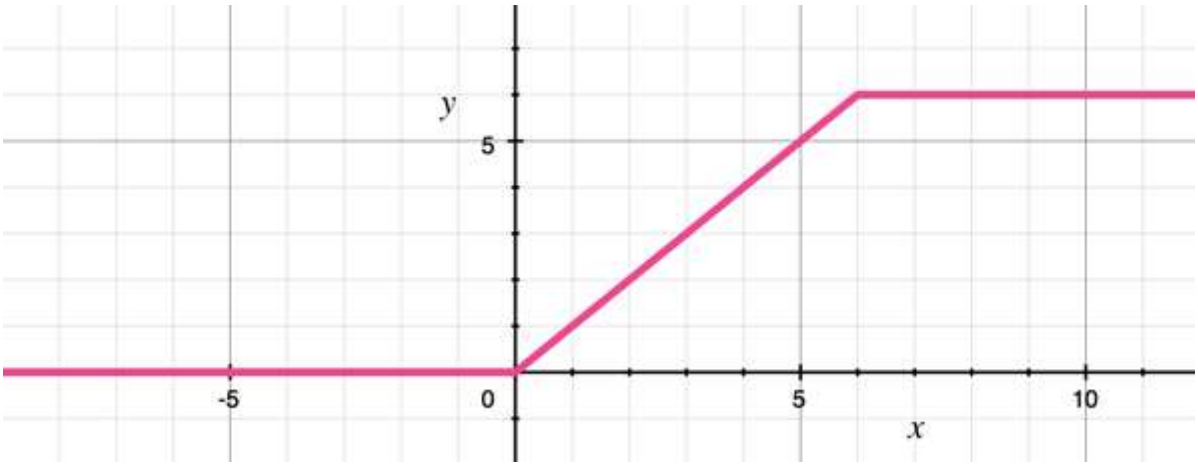

Рисунок 1.7 — Зображення функції ReLU6

Насправді існує більше ніж один MobileNet. Він був розроблений як сімейство нейромережевих архітектур. Існує кілька гіперпараметрів, які дозволяють грати з різними компромісами архітектури.

Найважливішим із цих гіперпараметрів є множник глибини, який заплутано також відомий як «множник ширини». Це змінює кількість каналів у кожному шарі. Використання множника глибини 0,5 зменшить удвічі кількість каналів, що використовуються в кожному шарі, що зменшує кількість обчислень на коефіцієнт 4, а кількість вивчених параметрів на коефіцієнт 3. Отже, це набагато швидше, ніж повна модель, але також менш точно.

Завдяки інновації глибоко відокремлених згортків, MobileNet повинен виконати приблизно в 9 разів менше роботи, ніж порівнянні нейронні мережі з однаковою точністю. Цей тип шарів працює настільки добре, що вдалося змусити моделі з 200+ шарами працювати в режимі реального часу, навіть на iPhone 6s.

MobileNet V2 все ще використовує згортки, що розділяються з глибиною, але його головний будівельний блок зображено на рисунку 1.8.

Наприклад, глибинний шар може працювати на тензорі зі 144 каналами, які завдяки проекційному шару згортаються лише до 24 каналів. Цей вид шару також називають шаром вузького мережі, оскільки він зменшує кількість

даних, що протікає по мережі. (Саме тут “залишковий блок вузького місця” отримує свою назву: вихід кожного блоку є вузьким місцем.)

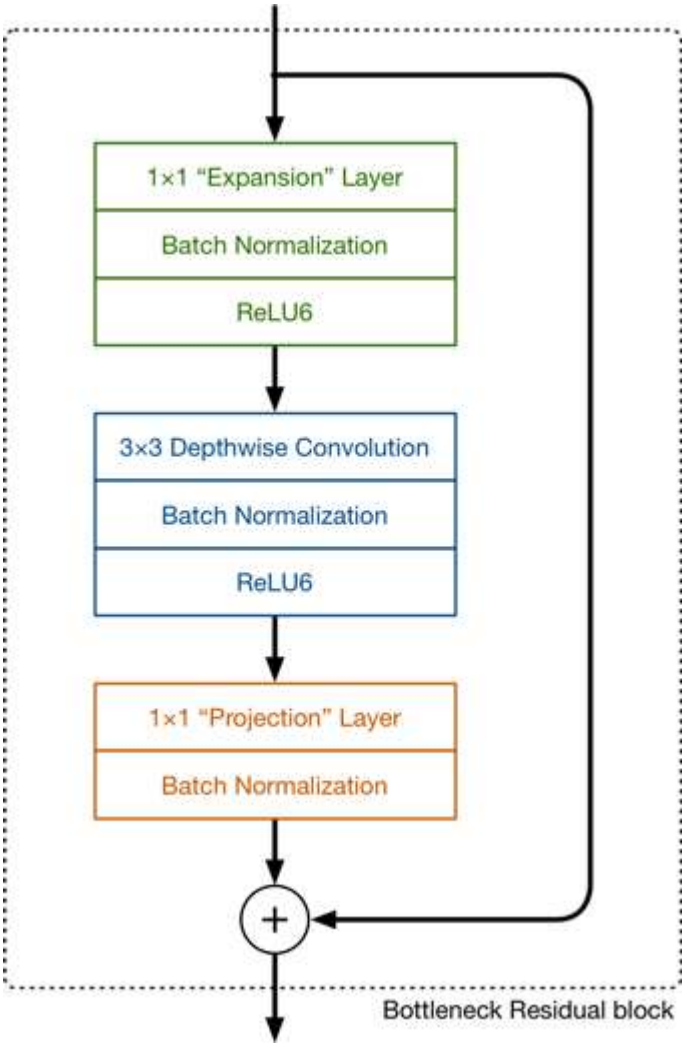

Рисунок 1.8 — Зображення головного блоку MobileNet V2

Перший шар – це новий елемент у блоці. Це також згортання 1 × 1. Його мета — розширити кількість каналів даних, перш ніж вони перейдуть до глибокої згортки. Отже, у цього шару розширення завжди присутнє більше вихідних каналів, ніж вхідних каналів – він значною мірою робить протилежний шар протилежний шару проекції.

Те, на скільки розширюються дані, визначає коефіцієнт розширення. Це один із тих гіперпараметрів для експериментів з різними компромісами архітектури. Фактор розширення за замовчуванням — 6.

Наприклад, якщо в блок є тензор з 24 каналами, шар розширення спочатку перетворює це в новий тензор з 24 * 6 = 144 каналами. Далі згортання поглиблень застосовує свої фільтри до 144-канального тензора. І нарешті, проекційний шар проектує 144 відфільтровані канали назад у меншу кількість, скажімо, 24, як показано на рисунку 1.9.

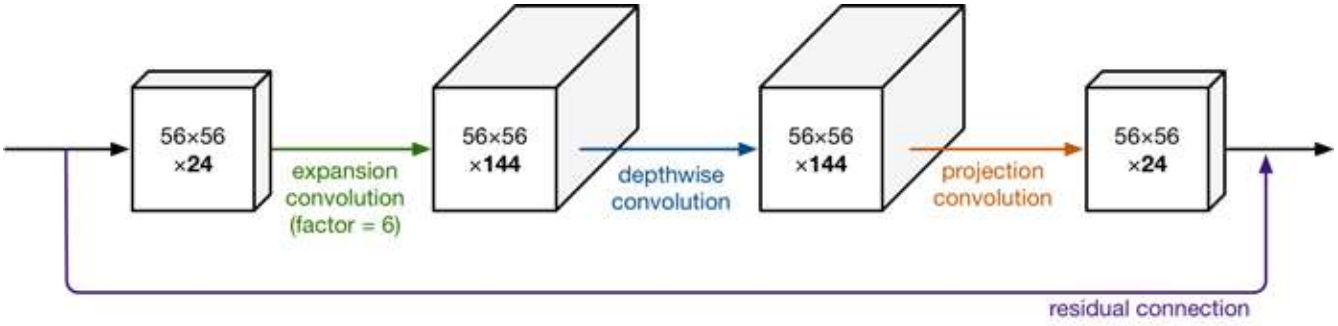

Рисунок 1.9 — Архітектура мережі MobileNet V2

Таким чином, вхід і вихід блоку – це тензори низьких розмірів, тоді як крок фільтрації, що відбувається всередині блоку, робиться за допомогою тензора досить великого розміру по очевидним причинам і несе поміжний характер.

Друга нова річ у будівельному блоці MobileNet V2 — це залишкове з’єднання. Це працює так само, як у ResNet, та існує, щоб допомогти з потоком градієнтів через мережу. (Залишкове з’єднання використовується лише тоді, коли кількість каналів, що входять до блоку, така ж, як кількість каналів, що виходять з нього, що не завжди так, оскільки кожні кілька блоків кількість вихідних каналів збільшується.)

Як і раніше, у кожного шару є пакетна нормалізація, а функцією активації є ReLU6. Однак на виході шару проекції немає функції активації, застосованої до нього. Оскільки цей шар створює дані низької розмірності, використання нелінійності після цього шару фактично не містить корисної інформації.

Даний тип мережі є зразком найбільш сучасної і передової мережі в сфері детектування зображень. Однак, з часом, через постійний розвиток сфери, навіть такі мережі, як MobileNet стають застарілими.

1.3.3 YOLO

Порівняно з іншими мережами класифікації пропозицій регіонів (Faster RCNN), які виконують виявлення в різних регіональних пропозиціях і, таким чином, виконують передбачення багаторазово для різних регіонів зображення, архітектура YOLO більше схожа на FCNN (повністю згорнута нейронна мережа) і передає зображення (n*n) один раз через FCNN та вихід — (m*m) прогнозування. Ця архітектура розділяє вхідне зображення на m*m сітку та для кожної генерації сітки 2 обмежувальних поля та вірогідності класу для цих зон [14].

Єдина згорткова мережа одночасно прогнозує кілька обмежувальних зон та ймовірності класів для цих зон. YOLO тренується на повних зображеннях і безпосередньо оптимізує продуктивність виявлення. Ця уніфікована модель має ряд переваг перед традиційними методами виявлення об’єктів. По-перше, YOLO надзвичайно швидка. Оскільки ми розглядаємо кадр як проблему регресії, нам не потрібен складний конвеєр. Ми просто запускаємо нашу нейронну мережу на нове зображення одноразово, щоб задетектувати об’єкти. Базова мережа працює зі швидкістю 45 кадрів в секунду без пакетної обробки на GPU Titan X, а швидка версія працює з більш ніж 150 кадрів в секунду. Це означає, що ми можемо обробляти потокове відео в режимі реального часу із затримкою менше 25 мілісекунд.

По-друге, YOLO глобально оцінює усе зображення під час прогнозування. На відміну від методів, що базуються на згорткових звуженнях та регіонах, YOLO бачить усе зображення під час тренувань та тестування, тому воно неявно кодує контекстну інформацію про об’єкти, а також їх зовнішній вигляд. Faster R-CNN, один з найкращих методів виявлення, помиляється через варіативність фонів для об’єктів, оскільки він не бачить

більшого контексту. YOLO робить менше половини кількості помилок фону порівняно з Faster R-CNN.

По-третє, YOLO узагальнює представлення предметів на зображенні. Під час навчання природніми зображеннями та тестуванні на художніх твірах YOLO перевершує найкращі методи виявлення, такі як DPM та R-CNN з великим запасом. Оскільки YOLO є дуже узагальнювальним алгоритмом, то є більш інваріативною до нових доменів або несподіваних входів.

Мережа використовує ознаки всього зображення для прогнозування кожного обмежувального поля та передбачує усі обмежувальні поля для всіх класів для зображення одночасно. Це означає, що мережа бере до уваги повне зображення та всі об'єкти на зображенні. Конструкція YOLO дозволяє проводити повну підготовку та швидкість в режимі реального часу, зберігаючи високу середню точність.

Система ділить вхідне зображення на сітку S × S. Якщо центр об'єкта потрапляє в комірку сітки, ця комірка відповідає за виявлення цього об'єкта. Кожна клітинка сітки прогнозує N обмежувальних полів та достовірність суми балів для цих зон. Ці показники достовірності відображають наскільки впевненою є модель у тому, що зона містить об'єкт, а також наскільки точною вона пропонує поле яке відводиться на цей об'єкт. Формально ми визначаємо довіру як Pr (Об'єкт) ∗ IOU. Якщо в цій комірці немає жодного об'єкта, показник довіри повинен бути нульовим. В іншому випадку ми хочемо, щоб показник довіри дорівнював перетину між союзом (IOU) між передбачуваним полем та основною істиною.

Кожне обмежувальне поле складається з 5 прогнозів: x, y, w, h та впевненості. Координати (x, y) являють собою центр вікна відносно меж комірки сітки. Ширина та висота прогнозуються відносно всього зображення. Нарешті, прогноз достовірності являє собою IOU між передбачуваним полем та будь-яким полем наземної істини. Кожна комірка сітки також прогнозує ймовірності C умовного класу, Pr (Classi | Object). Ці ймовірності обумовлені

в комірки сітки, що містить об’єкт. Ми прогнозуємо лише один набір ймовірностей класу на комірку сітки, незалежно від кількості зон N.

Основними перевагами Yolo є:

— швидкість (45 кадрів в секунду – краще, ніж у режимі реального часу);

— мережа розуміє узагальнене представлення об’єктів (Це дозволило навчати мережу на зображеннях реального світу, а передбачення на творах мистецтва були досить точними);

— більш швидка версія (з меншою архітектурою) – 155 кадрів в секунду, але менш точна;

— відкритий код.

Після успіху Yolo випустили другу версію, яка була ще краще і закривала недоліки першої. Але в наші дні є і третя версія даної мережі Yolo v3.

Саме третя версія взята за основу в даній магістерській кваліфікаційній роботі через набір характеристик та факту того, що вона є одною з найсучасніших мереж для детектування зображень. Усі принципи роботи мережі Yolo v3 буде розглянуто в подальших розділах кваліфікаційної роботи. В таблиці 1.1 наведено порівняльні характеристики усіх наведених архітектур.

Таблиця 1.1 – Порівняльна характеристика сучасних архітектур детектування

метод	швидкодія	точність	Залежність від к-ті об’єктів
Faster R-CNN	низька	висока	висока
SSD MobileNet	висока	низька	середня
Yolo	висока	середня	низька
Yolo v3	висока	висока	низька

Як видно з таблиці версія мережі Yolo v3 не має недоліків серед основних характеристик нейромереж.

1.4 Огляд відомих програмних реалізацій розпізнавання облич

Розпізнавання на обличчі вже стало гарячою темою 2019 року, але тепер, коли з’явилася технологія ідентифікації обличчя на сучасних смартфонах в реальному часі, розпізнавання обличчя стало ще більш популярною темою. Існує багато сучасних рішень на базі нейронних мереж.

Обслуговування клієнтів: Деякі банки Малайзії встановили системи, які використовують розпізнавання обличчя для виявлення цінних клієнтів банку, щоб банк міг надавати персоналізовану послугу. Таким чином, банки здатні отримувати більше доходів, утримуючи таких клієнтів і підтримуючи їх щасливими.

Сфера страхування: Багато страхових компаній використовують розпізнавання обличчя, щоб порівнювати обличчя людини з тими, які вказані в підтвердженні за зображенням. Таким чином, процес аутентифікації стає набагато швидшим.

Ось декілька з найбільш популярних додатків та програмних розширень:

Innovatrics SmartFace — це високоефективна, масштабована платформа сервера розпізнавання облич, здатна паралельно обробляти кілька потоків відео в реальному часі. Використовуючи провідний алгоритм Innovatrics, SmartFace дозволяє системним інтеграторам легко включати розпізнавання обличчя у свої рішення [15].

Швидке розгортання, без необхідних біометричних навичок. Інтерфейс додатку зображено на рисунку 1.10.

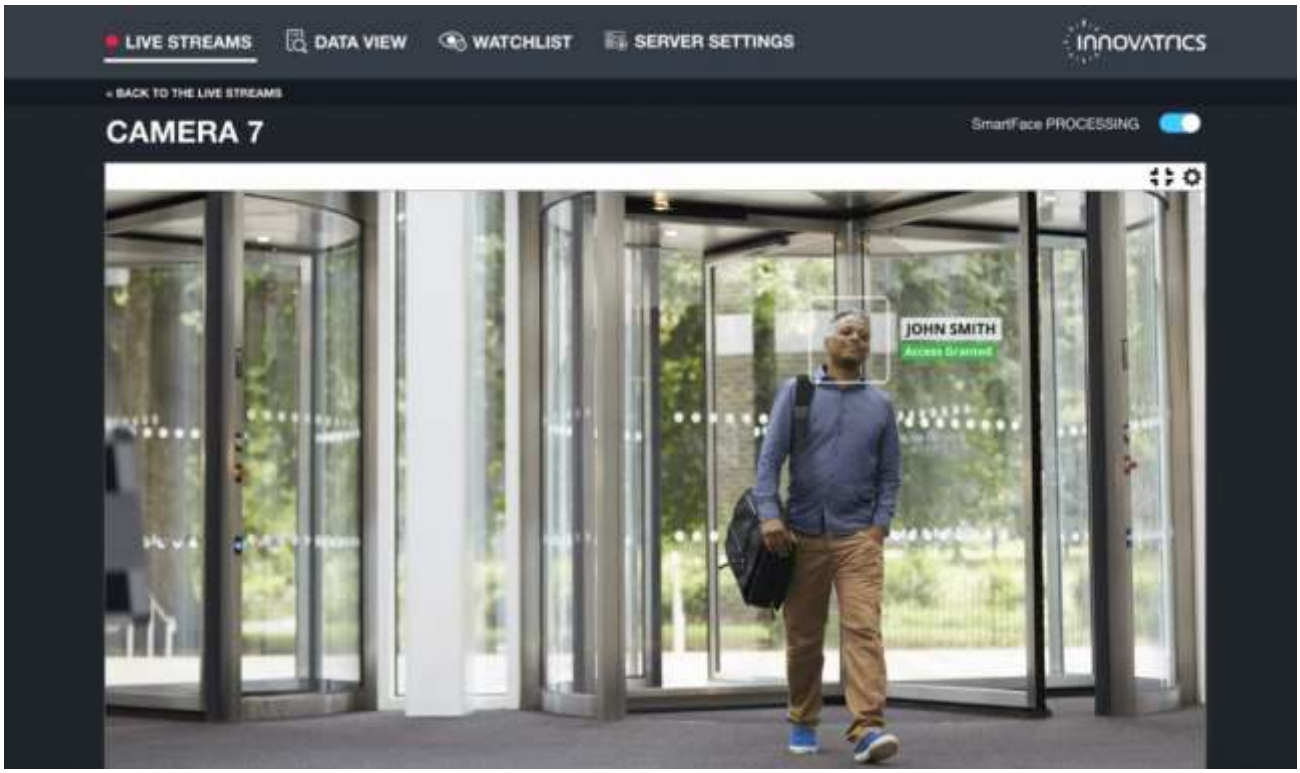

Рисунок 1.10 — Інтерфейс веб додатку SmartFace від Innovatrics

SmartFace вирішує типові проблеми, пов’язані з SDK розпізнавання обличчя, такі як відсутність спеціалізованих біометричних знань, тривалий час інтеграції та проблеми роботи з відеофайлами, безліч потоків IP-камер. SmartFace за своєю конструкцією легко інтегрувати, потенційно економлячи сотні людей-годин і значно скорочуючи періоди рентабельності інвестицій.

Kairos пропонує широкий спектр рішень розпізнавання зображень через їх API. Їх кінцеві точки API включають визначення статі, віку, емоційної глибини, розпізнавання обличчя як на зображенні, так і на відео, і багато іншого [16].

Один недолік у деяких API для розпізнавання осіб полягає в тому, що вони не здатні розрізняти обличчя та зображення обличчя. TrueFace.ai вирішує цю проблему своєю здатністю виконувати виявлення помилок через їх API. На рисунку 1.11 зображено основні функціональні вікна додатку.

Рисунок 1.11 — Інтерфейс основних екранів додатку на базі Kairos

Цей API створено користувачами в FaceX.io. FaceX API надає веб-сервісу виявлення облич і розпізнавання облич для вашого додатка або веб-сайту за допомогою декількох рядків коду. API FaceX може виявляти обличчя в завантажених зображеннях або URL-адресах. Інтегруйте цей API у свою програму, щоб мати змогу виконувати співставлення облич як пароль.

Однією цікавою функцією є те, що API має можливість виконувати «аналогічний пошук обличчя». Коли цій кінцевій точці API надається колекція облич і нове обличчя як запит API поверне колекцію подібних облич із колекції.

Цей API також використовує офлайн SDK для iOS і Android. Автономний SDK не забезпечує розпізнавання облич.

Face-Recognition — це відкритий репозиторій на github, який дає можливість кожному спробувати розпізнавати обличчя [17].

Він дозволяє розпізнавати та маніпулювати особами з Python або з командного рядка за допомогою найпростішої у світі бібліотеки розпізнавання облич.

Побудований за допомогою найсучаснішого розпізнавання обличчя dlib, побудованого за допомогою глибокого навчання. Модель має точність 98,38% за міткою Label Laces in Wild.

Це також забезпечує простий інструмент командного рядка face_recognition, який дозволяє розпізнавати обличчя у папці зображень з командного рядка. Основний екран показано на рисунку 1.12.

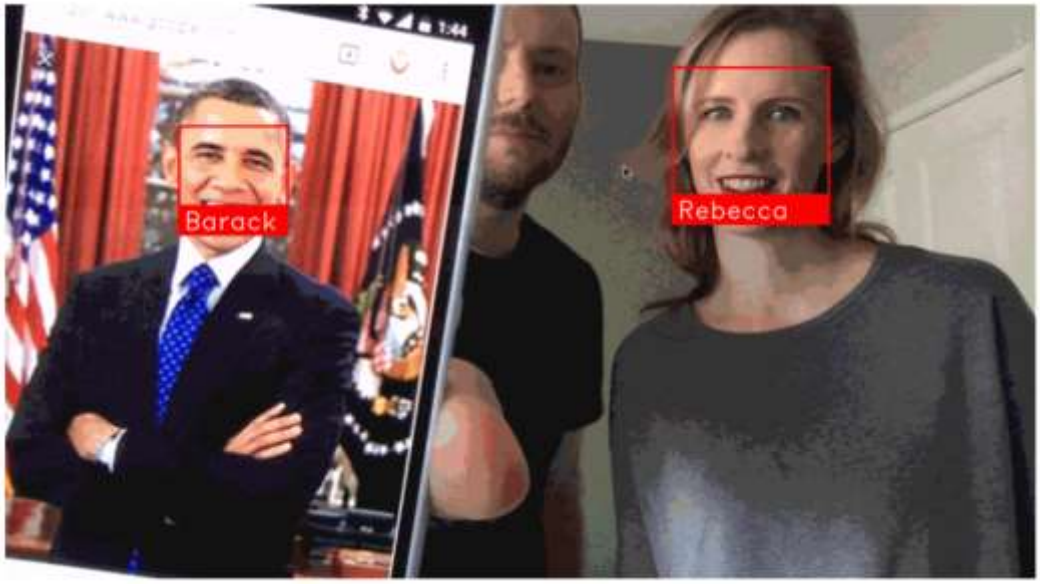

Рисунок 1.12 — Основне вікно демо версії відкритого face recognition

Усі наведені вище системи розроблялись в минулому, коли ще не було ні достатнього обладнання для виконання складних обчислень ні, сучасних методів, а також і такого стрімкого розвитку. Ці додатки і програмні засоби використовують алгоритми, минулих версій і зазвичай не намагаються витратити додаткові кошти на розробку систем з передовими технологіями, адже більшість з них зайняла деяку частину ринку технологій розпізнавання, і не втратили рентабельності.

Темою даної магістерської кваліфікаційної роботи є розробка програми детектування і розпізнавання облич використовуючи сучасні алгоритми згорткових нейронних мереж. Метою роботи є підвищення продуктивності сучасного алгоритму детектування.

1.5 Висновок

В даному розділі було описано основний принцип детектування і розпізнавання облич. Було приведено основні методи детектування об’єктів. Було проведено аналіз сучасних аглоритмів детектування, описано основні недоліки цих алгоритмів. Було наведено порівняльну характеристику засобів-аналогів, та проаналізовано їхні недоліки. Здійснено постановку задачі. Таким чином метою роботи є підвищення продуктивності детектування облич на зображеннях.

2 РОЗРОБКА ПРОЦЕСІВ ДЕТЕКТУВАННЯ ТА РОЗПІЗНАВАННЯ ОБЛИЧ НА ЗОБРАЖЕННЯХ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

2.1 Розробка архітектури згорткової нейромережі детектування облич на зображеннях

За основу детектування взято модель YOLOv3. Ідеї основних алгоритмів даної мережі взяті з найкращих робіт і поєднані в одне ціле. Це зробило дану систем однією з найкращих систем для детектування в наші дні.

Після попередньої версії алгоритму YOLO9000 система прогнозує обмеження зон детекції, що використовують кластери розмірів як зони для прив'язки, які ще називають якоря. Мережа прогнозує 4 координати для кожного обмежувального вікна, tx, ty, tw, th. Якщо комірка зміщена відносно верхнього лівого кута поля зображення на (cx, cy) та обмежувальне поле має ширину та висота pw, ph, то прогнози відповідають формулам 2.1:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{2.1}$$

Під час навчання ми використовуємо суму квадрату помилки. Якщо істинне значення для деяких координат прогнозування — це $\hat{t}_*$ градієнт — значення істини (обчислюється з істинним значення) мінус наш прогноз: $\hat{t}_* - t_*$. Ця істинне значення можна легко обчислити, інвертуванням рівняння вище.

YOLOv3 прогнозує показник об'єктивності для кожного зони де може бути об'єкт з використанням логічної регресії. Це повинно бути 1, якщо зона

детекції попередньо перекриває істинну зону детектування ніж будь-яке інше поле детектування раніше. Якщо обмежувальне не найкраще, але перекриває наземний об'єкт істини на більше ніж деякий поріг ми ігноруємо передбачення, слідуючи. Ми використовуємо поріг 0.5. Ширина та висота зони прогнозується, як зміщення від кластерних центроїдів. Відбувається прогнозування центральних координат зони відносно місця розташування, яке фільтрується алгоритмом за допомогою функції сигмоїди.

На відміну від попередньої мережі, алгоритм призначає лише одне обмежувальне поле для кожної істини об’єкта. Якщо попередня зона не відповідає істинності, функція втрат не діє для координатних чи класових передбаченнях, лише зменшується об’єктивність.

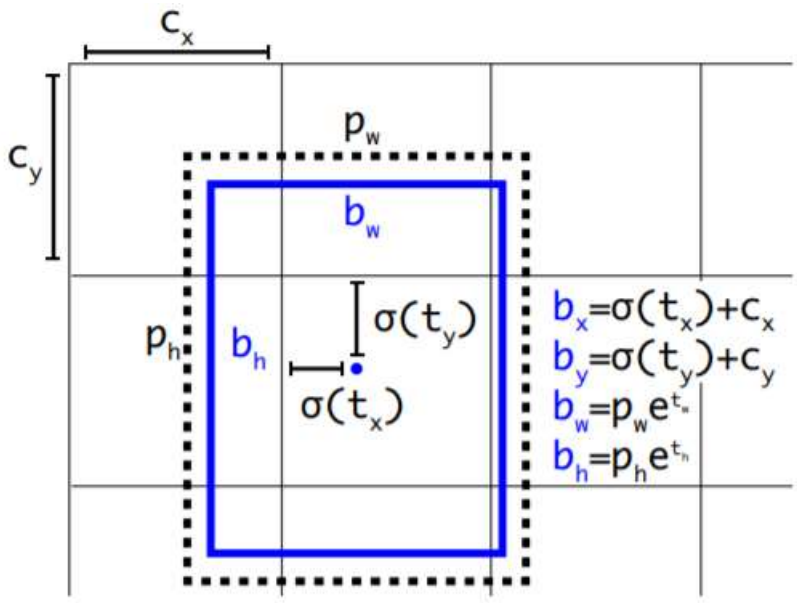

Рисунок 2.1 — Зони детекції з пріоритетом і передбаченням локації

Кожне поле передбачає класи, обмежувальні поля які можуть там міститися, з використанням багатозначної класифікації. Softmax тут не використовується тому, що це непотрібно для точної детекції, натомість просто використовуються незалежні логістичні класифікатори. Під час навчання використовується бінарна крос-ентропія втрати для прогнозування класу.

Цей метод допомагає, коли ми переходимо до більш складних даних, що є звичайною справою для датасетів з відкритими зображеннями. У таких наборах даних є багато міток, що перекривають один одного (наприклад жінка та людина). Використання softmax нав'язує припущення, що кожне поле відповідає одному класу, що часто не так. Мультикласовий підхід краще підходить для моделювання даних.

YOLOv3 прогнозує зони в троьох різних розширеннях. Система витягує ознаки з цих розширень, використовуючи аналогічну концепцію для ознак, як у пірамідальних мереж. З нашого базового екстрактора функцій додаємо кілька згорткових шарів. Останній з них передбачає тривимірне тензорне кодування зони детектування, відповідність до об’єкту та передбачення для класу. У ході експериментів з даними COCO прогнозуємо 3 зони в кожному вимірі, це тензор $N \times N \times [3 * (4 + 1 + 80)]$ для 4-х зсувів рамки, 1 об'єктивність прогнозування та прогнозування на 80 класів. Далі ми беремо карту особливостей з двох попередніх шарів та збільшити вибірку на 2. Ми також беремо карту характеристик з попередньої версії та об'єднуємо з нашими пірамідальними ознаками, використовуючи конкатенацію. Цей метод дозволяє отримати більш змістовну семантична інформацію з вищезгаданих ознак і деяку інформацію з попередньої карти ознак. Потім ми додаємо ще кілька згорткових шарів для обробки цієї комбінованої карти ознак і врешті-решт прогнозуємо а схожий тензор, хоча він вже вдвічі більший. Ми виконуємо ту саму дію ще раз, щоб передбачити поля для остаточні оцінки. Таким чином, наші прогнози для 3-го розширення мають усю інформацію про усі попередні результати обчислень, як із з самих верхніх шарів мережі.

Використовується кластеризація k-means для визначення наших обмежувальних полів. Ми обчислюємо 9 кластерів і ще 3 масштабуються довільно, а потім розподіляються рівномірно відносно усіх розмірів об’єктів в наборі даних.

YOLO використовує квадрат помилки між прогнозами та істиною для обчислення втрат. Функція втрат складається з:

— класифікаційні втрати;

— втрата локалізації (помилки між передбачуваним полем границі та істиною);

— втрата впевненості (об'єктивність зони прогнозування).

Якщо об'єкт виявлений, втрата класифікації в кожній комірці - це помилка квадрата умовної ймовірності класу для кожного класу:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \tag{2.2}$$

де $\mathbb{1}_i^{obj} = 1$ якщо об’єкт з’явився в клітинці *i*, інакше 0,

$\hat{p}_i(c)$ позначає умовну вірогідність класу *c* в клітинці *i*.

Функція втрати локалізації вимірює помилки у передбачуваних місцях та розмірах зон. Береться до уваги тільки лише поле, відповідальне за виявлення об'єкта, як показано в формулі 2.3.

$$\begin{aligned} &\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \end{aligned} \tag{2.3}$$

де $\mathbb{1}_{ij}^{obj} = 1$ якщо *j* зона детектування в клітинці *i* відповідає за детектування, інакше 0,

λ_{coord} збільшує ваги через функцію втрат в зоні детектування.

Навмисно нерівномірно зважуються абсолютні помилки у великих і маленьких зонах детектування, тобто, щоб не було ситуації, коли помилка у 2 пікселі у великій зоні однакова для невеликої зони. Щоб частково вирішити це, YOLO передбачає квадратний корінь ширини та висоти обмежувальної

зони замість звичайних ширини та висоти. Крім того, щоб зробити акцент на точності граничного поля, помножиться функція втрати на λcoord (за замовчуванням: 5).

Якщо в полі виявлено об'єкт, функція втрати довіри (вимірювання об'єктивності поля) виглядає так:

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(C_i-\hat{C}_i\right)^2 \quad , \tag{2.3}$$

де $\hat{C}_i$ впевненість зони *j* в в клітинці *i*,

$\mathbb{1}_{ij}^{obj}=1$ якщо *j* зона відповідає розпізнаному об’єкту, інакше 0.

Якщо в полі не виявлено об'єкт, функція втрати довіри виглядає так:

$$\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_i\right)^2 \quad , \tag{2.4}$$

де $\mathbb{1}_{ij}^{noobj}$ доповнення $\mathbb{1}_{ij}^{obj}=1$,

$\hat{C}_i$ впевненість зони *j* в клітинці *i*,

λ_{noobj} ваги, які знижують функцію втрати при детектування фону.

Більшість зон детектування не містить жодних об'єктів. Це спричиняє проблему дисбалансу класів, тобто ми навчаємо модель виявляти фон частіше, ніж виявляти об'єкти. Щоб виправити це, ми зменшуємо цю втрату на коефіцієнт λnoobj (за замовчуванням: 0,5).

Остаточний функція втрат додає функцію локалізації, функцію впевненості та функцію класифікації разом:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
\tag{2.5}
$$

YOLO може робити повторювані виявлення для одного і того ж об'єкта. Щоб виправити це, YOLO застосовує не максимальне об’єднання для видалення дублікатів з меншою надійністю. Немаксимальне об’єднання додає 2- 3% у mAP.

Ось одна з можливих реалізацій максимального об’єднання:

— Сортувати прогнози за достовірністю;

— починати з найкращих показників, ігнорувати будь-який поточний прогноз, якщо ми знаходимо попередні прогнози, які мають той самий клас та IoU> 0,5 з поточним прогнозом;

— Повторити крок 2, поки не будуть встановлені всі прогнози.

Тренування відбувається на повних зображеннях без важкого негативного впливу на якість або будь-якого іншого перетворення. Використовується багатомасштабне навчання, багато розмноження даних, нормалізація набору зображень, та всі інші зміни на зображенні, що змінюють розмір навчальних даних.

Дана мережа має високу точність, але нам потрібно знайти методи зменшення кількості обчислень. Для цього відсікаються усі лишні шари згортки та останній шар Yolo. Дана архітектура отримала назву Yolov3 tiny. Шари нової моделі відображено в таблиці 2.1.

Таблиця 2.1 – Представлення усіх шарів компактної архітектури

Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	Maxpool		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	Convolutional	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32
3	Maxpool		2 × 2/2	208 × 208 × 32	104 × 104 × 32
4	Convolutional	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64
5	Maxpool		2 × 2/2	104 × 104 × 64	52 × 52 × 64
6	Convolutional	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128
7	Maxpool		2 × 2/2	52 × 52 × 128	26 × 26 × 128
8	Convolutional	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256
9	Maxpool		2 × 2/2	26 × 26 × 256	13 × 13 × 256
10	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
11	Maxpool		2 × 2/1	13 × 13 × 512	13 × 13 × 512
12	Convolutional	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
13	Convolutional	256	1 × 1/1	13 × 13 × 1024	13 × 13 × 256
14	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
15	Convolutional	255	1 × 1/1	13 × 13 × 512	13 × 13 × 255
16	YOLO				
17	**Route 13**				
18	Convolutional	128	1 × 1/1	13 × 13 × 256	13 × 13 × 128
19	Up-sampling		2 × 2/1	13 × 13 × 128	26 × 26 × 128
20	**Route 19 8**				
21	Convolutional	256	3 × 3/1	13 × 13 × 384	13 × 13 × 256
22	Convolutional	255	1 × 1/1	13 × 13 × 256	13 × 13 × 256
23	YOLO				

Таким чином ми позбуваємося непотрібних обчислень так, як об’єктом детектування будуть лише обличчя достатнього розміру, вистачить меншої кількості шарів та пірамідальні ознаки лише 2 просторів.

2.2 Математична модель відсікання надлишкових фільтрів

Розробка глибоких згорткових нейронних мереж (CNN) у багатьох реальних програмах значною мірою затрудняється через їх високу обчислювальну вартість.

В основі даної магістерської кваліфікаційної роботи лежить ідея про те, що не всі нейрони в шарах, а точніше фільтри в згорткових шарах, впливають

на результат детектування. Деякі ваги в шарах несуттєво змінюють значення на подальших нейронах, але обчислювальні ресурси на обрахування даних вузлів витрачаються. Якщо прибрати залишкові вузли можна зменшити кількість обчислювальних операцій, тим самим покращити два показники:

— Зменшити час на обчислення, за рахунок зменшення кількості, операцій множення і додавання всередині згорткових шарів;

— Зменшити розмір вихідних ваг мережі, тим самим зменшити кількість пам’яті, яка виділяється для виконання їх в пам’яті.

В багатьох роботах пропонується стискання великих CNN або безпосередньо навчатись виконувати обчислення з меншою кількістю ітерацій. До них належать апроксимація на низькому рівні, мережеве квантування та бінаризація ваг згортки, динамічне виведення, тощо. Однак більшість з цих методів вирішує одну проблему, в той час як з’являється інша. Більше того, деякі методи вимагають спеціально розроблені програмно-апаратні прискорювачі для прискорення виконання.

На рисунку 2.2 зображено основну схему відсікання, з початковим та кінцевим виглядом мережі.

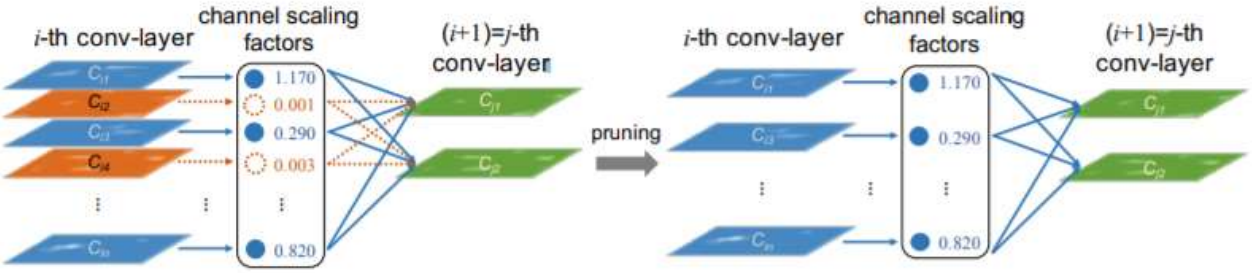

Рисунок 2.2 — Зображення основної ідеї відсікання надлишкових шарів

Запропоновано просту схему для досягнення розрідженості каналів у глибоких CNN. У цьому розділі ми спочатку обговоримо переваги та труднощі відсікання на рівні каналів мережі та представимо, як ми використовуємо шари масштабування в наборі нормалізації, щоб ефективно ідентифікувати та відсікти неважливі канали в мережі.

Відсікання може бути реалізовано на різних рівнях мережі, наприклад, рівень ваги, рівень ядра, рівень каналу або рівень-абстракції.

Дрібнозернистий рівень (наприклад, рівень ваги) має найбільшу гнучкість і рівень компресії, але вона зазвичай вимагає спеціального програмного забезпечення або апаратні прискорювачі для швидкого виконання на розріджених моделях. Навпаки, для основних шарів не потрібні багато обчислювальних потужностей, при цьому вони є менш гнучкими, в той час як деякі цілі другорядні шари потрібно обрізати повністю.

Насправді видалення шарів ефективно лише тоді, коли глибина достатньо велика, наприклад, більше 50 шарів. Для порівняння, розрідженість на рівні каналу забезпечує хороший компроміс між гнучкістю та простотою реалізації.

Цей метод може бути застосований до будь-яких типових мереж CNN або повністю з’єднаних мереж (кожен нейрон як канал, тоді являється каналом) тоді отримана мережа по суті є "стоншеною" версією необрізаної мережі, які ефективно використовуються на різних платформах і змаганнях присвячених CNN.

Досягнення розрідженості на рівні каналу вимагає обрізка всіх вхідних та вихідних з'єднань, пов'язаних із каналом. Це робить метод безпосередньої обрізка ваг за попередньо підготовленою моделлю неефективним, так як всі ваги на вхідному або вихідному кінці каналу, буде мати майже нульові значення. Як повідомляється в попередніх дослідженнях, обрізка каналів на попередньо підготовлених ResNets може призвести лише до зменшення кількості параметрів на 10% кількості без втрати точності. Вирішити проблему можна шляхом примусового регулювання обмеженості в навчальних шарах. Зокрема, вони адаптують групу LASSO, щоб підігнати ваги фільтрів відповідним каналам до нуля під час тренувань. Однак такий підхід вимагає обчислення градієнтів додаткового шару регуляризації щодо всіх ваг фільтра, що є нетривіальним. Ми вводимо просту ідею для вирішення вищесказаних проблем.

Ідея полягає у вводі коефіцієнту масштабування γ для кожного каналу, який помножено на вихід цього каналу. Тоді можна спільно навчати мережеві ваги та ці коефіцієнти масштабування, з регулюванням розрідженості.

В кінці ми відсічемо шари по несуттєвим факторам і проведемо точне налаштування ваг вузлів, що залишились.

Зокрема, навчальна мета нашої підхід описується як:

$$L = \sum_{(x,y)} l(f(x,W),y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma) \tag{2.6}$$

,

де *(x, y)* позначають вхід і мету тренування,

W позначає треновані ваги,

перша сума відповідає рівню функції втрати тренувань CNN,

g (·) - штраф за коефіцієнтами масштабування, а λ врівноважує два правила.

У нашому експерименті ми вибираємо g (s) = | s |, який відомий як L1-норма і широко використовується для досягнення розрідженості.

Суміжний градієнт прийнятий як метод оптимізації для штрафної умови L1, без втрат. Альтернативою буде заміна покарання L1 з гладким покаранням L1, яке слід уникати використовуючи суміжний градієнт у неявній точці.

Оскільки відсікання каналу по суті відповідає видаленню всіх вхідних та вихідних з'єднань цього каналу, ми можемо безпосередньо отримати звужену мережу, як на рисунку 2.2 не вдаючись до будь-яких спеціальних операцій з розрідженими обчисленнями. Фактори масштабування виступають агентами для вибору каналу. Оскільки вони спільно оптимізовані з мережевими вагами, мережа може автоматично ідентифікувати незначні канали, які можна безперешкодно видалити, що впливає на ефективність узагальнення.

Нормалізація набору даних була прийнята більшістю сучасних CNN як стандартний z_{in} підхід для досягнення швидкої $\mathcal{B}$ конвергенції та

кращих показників узагальнення. Те, як такий підхід нормалізує активацію змушує теж використати простий та ефективний метод, який би враховував канальні фактори масштабування. Зокрема, шар який відповідає за нормалізацію, нормалізує внутрішні активації, використовуючи статистику, об'єднаного набору. Припустимо, що і входи шару нормалізації, це поточний міні набір і нормалізація набору виконує такі трансформації:

$$\hat{z} = \frac{z_{in} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}; \;\; z_{out} = \gamma \hat{z} + \beta \quad , \tag{2.7}$$

де $\mu_{\mathcal{B}}$ та $\sigma_{\mathcal{B}}$ значення відхилень від активації, на $\mathcal{B}$,

γ та β здатні для налаштування параметри афінних перетворень, що надають можливість лінійно перетворювати активації назад до будь-яких розширень.

Загальноприйнятою практикою є вставляння шару нормалізації набору після згорткового шару з параметрами масштабування або зміщення каналу. Тому ми можемо безпосередньо використовувати γ параметри в шарі нормалізації, як фактори масштабування, необхідні для мережевого відсікання.

Він має велику перевагу, якщо не вводити накладні витрати на мережу. Насправді, це, мабуть, і найефективніший спосіб, коли ми можемо дізнатися значущі коефіцієнти масштабування для відсікання каналів.

1) якщо ми додамо шари масштабування до CNN без шару нормалізації, значення коефіцієнтів масштабування не мають значення для оцінки важливості каналу, оскільки обидва згорткові і шари масштабування - це лінійні перетворення. Отримати ті самі результати можна, зменшивши значення коефіцієнта масштабування під час оновлення ваг у згорткових шарах.

2) якщо розмістити шар масштабування раніше шару нормалізації, ефект масштабування шару масштабування буде повністю скасований процесом нормалізації.

3) якщо ми вставимо шар масштабування після шару нормалізації, є два послідовні коефіцієнти масштабування для кожного каналу.

Після тренування на рівні каналу зменшилась регуляризація, і ми отримаємо, модель, в якій багато коефіцієнтів масштабування перетворюються в нуль. Тоді ми можемо відсікти канали з майже нульовим масштабуванням факторів, видаляючи всі їх вхідні та вихідні з'єднання та відповідні ваги.

Обрізаємо канали із загальним порогом для всіх шарів, який визначено як певний відсоток усіх значень коефіцієнта масштабування. Наприклад, ми відсікаємо 70% каналів із меншими коефіцієнтами масштабування вибираючи відсотковий поріг як 70%. Роблячи це, ми отримуємо більш компактну мережу з меншими параметрами і кількістю оперативної пам'яті, що необхідна, щоб тримати ваги в пам’яті, а також менше обчислювальних операцій.

Відсікання може тимчасово призвести до певної втрати точності, коли коефіцієнт відсікання високий. Але це може бути значною мірою компенсовано наступним процесом тонкого налаштування на обрізаних шарах мережі. Деякі експерменти показали, що тонко налаштована вузька мережа може навіть досягти вищої точності, ніж оригінальна мережа у більшості випадках.

Ми також можемо повторити запропонований метод із схеми навчання однопрохідного навчання (навчання с регуляризація рідкості, відсічення та тонке налаштування). Зокрема, мережева процедура відсічення призводить до звуження мережі, до якої ми могли б знову застосувати всю процедуру навчання, щоб навчитися ще більш компактні моделі. Це проілюстровано пунктирною лінією на рисунку 2.3. Експериментальні результати показують, що ця багатопрохідна схема може призвести до ще кращих результатів в плані швидкості виконання.

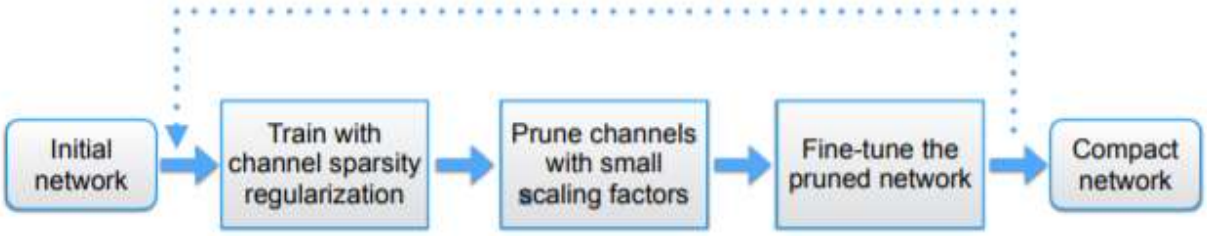

Рисунок 2.3 — Зображення процесу циклічного відсікання і налаштування

Фінальні результати зміни архітектури представленні на рисунку 2.4.

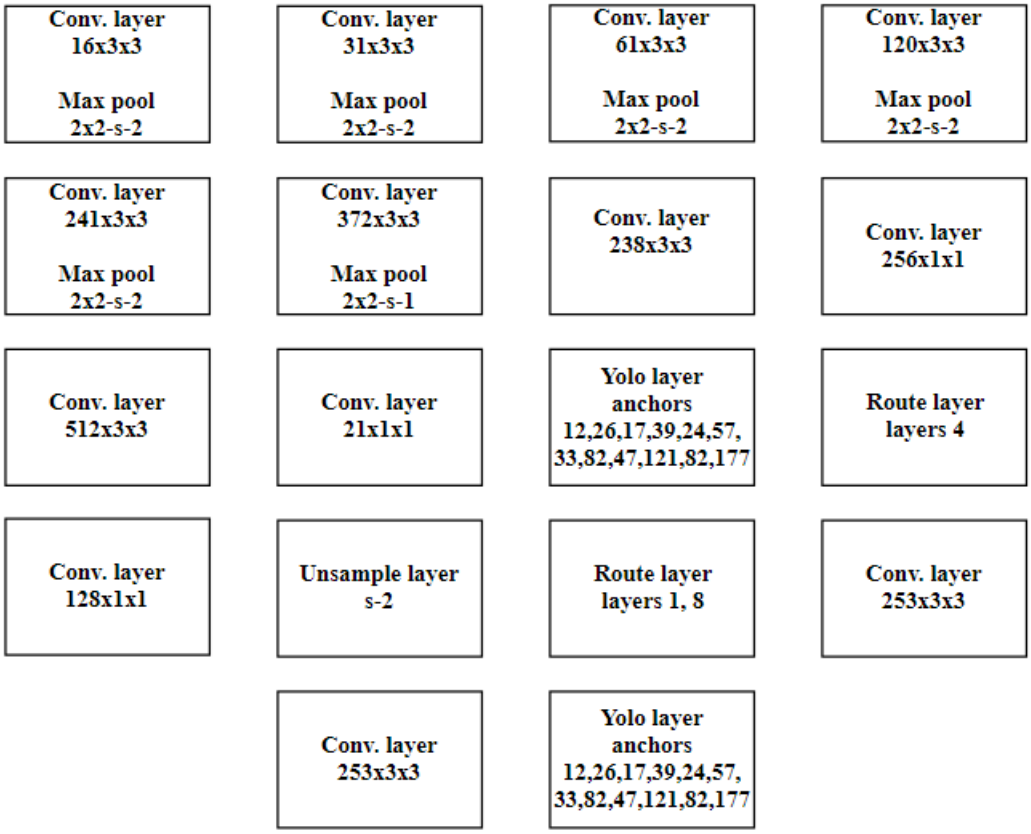

Рисунок 2.4 — Архітектура розробленої нейромережі

Кількість шарів мережі склала 24. Це набагато менше чим в третій версії оригінальної Yolo.

Окрім зміни кількості фільтрів також було зменшено кількість якорів облич, що зменшить кількість обрахунків положення об’єктів на зображенні,

але не призведе до втрати якості, через те, що відносність сторін оригінальної мережі набагато різноманітніша, адже вона була навчена на 80 абсолютно різних по формі об’єктів. Обличчя має майже однакові на всіх зображеннях відносні ширину і висоту, отже необхідність обраховувати аж 9 якорів не є необхідно. Їх кількість було зменшено до 6.

2.3 Розробка методу розпізнавання облич

Система розпізнавання обличчя — це технологія, здатна ідентифікувати або перевірити людину з цифрового зображення або відеокадру з джерела відео. Існує кілька методів, за допомогою яких працюють системи розпізнавання обличчя, але в цілому вони працюють, порівнюючи вибрані риси обличчя із заданого зображення з обличчями в базі даних. Він також описується як програма на основі біометричного штучного інтелекту, яка дозволяє однозначно ідентифікувати людину, аналізуючи шаблони, засновані на фактурі та формі обличчя людини.

Розпізнавання обличчя використовується в багатьох підприємствах. Деякі із застосувань системи розпізнавання облич — це розблокування телефонів, розумна реклама, тегування людей на платформах соціальних медіа, діагностика захворювань, відстеження активності, перевірка платежів тощо.

Розпізнавання, перевірку та ідентифікацію обличчя часто плутають. Розпізнавання обличчя — це загальна тема, яка включає як ідентифікацію обличчя, так і перевірку обличчя (також називається автентифікацією). З одного боку, перевірка обличчя пов'язана з підтвердженням заявленої ідентичності на основі зображення обличчя та прийняттям або відхиленням претензії на особу (відповідність один до одного). З іншого боку, метою ідентифікації обличчя є виявлення людини на основі зображення обличчя. Це зображення обличчя має порівнюватися з усіма зареєстрованими особами (одна на багато співпадаючих) у базі даних.

Для генерації ознак обличчя використовуємо реалізацію OpenFace, яка використовує архітектуру FaceNet Google, яка дає кращий результат із використанням бібліотеки dlib.

Коли ми ізолюємо обличчя від зображення, обличчя людини може проектуватися будь-яким способом або людське обличчя може розміщуватись у просторі по різному. Коли ми створюємо вбудовування ізольованого обличчя, вкладення обличчя однієї і тієї ж людини, спроектоване різними способами, може різнитися в багатьох варіантах.

Ми повинні повернути кожне зображення так, щоб очі і губи завжди були на зразковому місці зображення. Це полегшить нам порівняння облич. Для цього ми будемо використовувати алгоритм, який називається оцінкою орієнтиру обличчя.

Основна ідея полягає в тому, що ми створимо 512 конкретних ознак (які називаються орієнтирами), які існують на кожному обличчі. Тоді ми навчимо алгоритм машинного навчання, щоб ми могли знайти ці 512 конкретних ознак на будь-якому обличчі, як видно на рисунку 2.5.

Тепер ми можемо визначити ці орієнтири на кожному обличчі, ми будемо просто обертати, масштабувати і зрізати зображення, щоб очі і рот були максимально добре зосереджені в центрі.

Після того, як ми виділимо зображення з фону і попередньо обробимо його за допомогою нейронної мережі нам потрібно знайти спосіб зобразити обличчя в числовій вставці. Ми можемо представити його за допомогою перевіреної глибокої нейронної мережі OpenFace.

Під час навчальної частини алгоритму OpenFace через нейронну мережу проходить 500 000 зображень. OpenFace навчає ці зображення, щоб створити 512 вкладень на обличчя, які представляють загальне обличчя. OpenFace використовує архітектуру FaceNet Google для вилучення функцій і використовує функцію потрійної втрати, щоб перевірити, наскільки точно нейронна сітка класифікує обличчя.

Нейронну мережу потрібно навчити таким чином, щоб вбудовування зображення якоря та позитивного зображення було подібним, а вбудовування зображення якоря та негативного зображення повинні бути сильно різними.

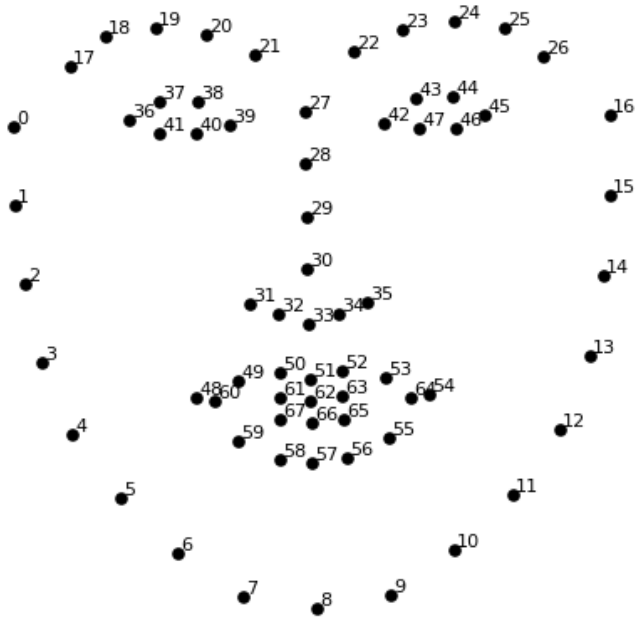

Рисунок 2.5 — Приклад 64 ознак обличчя

Для вивчення параметрів нейронної мережі нам потрібно тренувати на кожному з трьох різних наборах зображень одне відоме зображення обличчя, яке називається зображенням якоря, інше зображення тієї ж людини, як зображення якоря, яке називається позитивним зображенням, а третє зображення іншої людини, ніж зображення якоря, яке називається негативним зображенням .

Distance b/w anchor embedding & positive embedding = $||F(A)-F(P)||^2$

Distance b/w anchor embedding & negative embedding =$||F(A)-F(N)||^2$

Рівність, $||F(A)-F(P)||^2 — ||F(A)-F(N)||^2 \leq 0$

Треба, щоб наш dist (A, P) був меншим, а dist (A, N) вищим. Щоб переконатися, що NN не просто приводить dist (A, P) & dist (A, N) дорівнює нулю, щоб задовольнити рівняння, тому ми зменшуємо RHS з 0 до меншої граничної величини (α).

Тепер рівність зводиться до , $||F(A)-F(P)||^2 — ||F(A)-F(N)||^2 \leq -\alpha$

$||F(A)-F(P)||^2 — ||F(A)-F(N)||^2 \leq -\alpha$

Для N набору триплетних зображень:

$$Loss = \sum_{i=1}^{N} \left[\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right]_+$$

(2.8)

Але після того, як мережа пройшла навчання, вона може генерувати вимірювання для будь-якого обличчя, навіть такого, якого вона ніколи не бачила!

Потрібно запустити наші образи обличчя через їх попередньо навчену мережу з відкритим обличчям, щоб отримати 512 вимірювань для кожного обличчя. Вбудовування — це загальне зображення для будь-кого обличчя. На відміну від інших зображень обличчя, це вбудовування має особливу властивість, що більша відстань між двома наборами характеристик обличчя означає, що обличчя, ймовірно, не однієї особи.

Це властивість полегшує завдання кластеризації, виявлення подібності та класифікації легше, ніж інші методи розпізнавання обличчя, де евклідова відстань між ознаками не має сенсу.

512-мірне вбудовування, повернене моделлю FaceNet, може бути ефективно використане для кластеризації облич, оскільки обговорене вище відстань вбудовування буде ближчим для подібних граней і далі для не схожих граней.

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

(2.9)

Щоб порівняти два зображення, треба створити вбудовування для обох зображень, подаючи окремо через модель. Тоді ми можемо використовувати евклідову відстань, щоб знайти відстань, яка буде меншою величиною для подібних граней і більшою величиною для різних граней.

Для кожного зображення запиту ми обчислюємо евклідову відстань між вбудуванням зображення запиту та вбудуванням кожного із зображень, присутніх у наборі даних. Зображення, що має найменшу евклідову відстань, може розглядатися як зображення тієї самої людини, як у зображенні запиту.

Тепер ми вбудуємо кожне із зображень у набір даних, ми можемо навчити класифікатор, який приймає обличчя як дані про поїзд та імена як мітку класу даних навчання. Навчити модель машинного навчання за допомогою різних методик та налаштувати гіперпараметр кожної з моделей, щоб отримати найкращу точність, можна за допомогою відповідних інструментів.

2.4 Висновок

У даному розділі було розроблено архітектуру згорткової мережі для детектування та розпізнавання облич на зображеннях з використанням згорткових нейронних мереж та математичну модель відсікання лишніх фільтрів мережі, а також допоміжних засобів для покращення точності детектування облич. Описано методи, що можуть бути використані для детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж. Було розроблено метод розпізнавання облич в програми детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж.

3 ПРОЕКТУВАННЯ СИСТЕМИ ДЕТЕКТУВАННЯ ТА РОЗПІЗНАВАННЯ ОБЛИЧ НА ЗОБРАЖЕННЯХ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

3.1 Обґрунтування вибору мови програмування

Здійснимо вибір мови програмування для розробки системи розпізнавання об‘єктів на льоту з використанням дерева рішень.

Python (найчастіше вживане прочитання — інтерпретована об'єктно-орієнтованамова програмування високого рівня зі строгою динамічною типізацією [18].

Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах.

В мові програмування Python підтримується кілька парадигм програмування: об'єктно
орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

чистий синтаксис (для виділення блоків слід використовувати відступи);

- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ [19].

Інтерпретатор мови Python і багата Стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Порівняння мов Python і C++ зображено в таблиці 3.1.

Враховуючи всі переваги і недоліки, була вибрана мова Python, адже вона використовується в комп'ютерному зорі і має безліч бібліотек для машинного навчання. Саме через простоту застосування, і в сфері комп’ютерного зору і нейромережевих методів застосовують пайтон. Одна дослідницька ітерація триває набагато менше, коли в цих ітераціях, які не є фінальними застосовують мову програмування, яка потребує мінімум зусиль і часу.

Таблиця 3.1 – Порівняння мов програмування Python і C++

Ознака порівняння	Python	C++
Підтримка методології ООП	Так	Так
Кроссплатформеність	Так	Ні
Наявність контейнерів (List, Set, Map)	Так	Ні, але вони подібні
Багато безкоштовних бібліотек	Так	Так
Сумісність різних версій	Так	Ні
Відкритий доступ до вихідного коду	Так	Так
Збірник сміття	Так	Ні

3.2 Особливості середовища PyCharm

PyCharm — інтегроване середовище для розробки додатків, спеціально для мови програмування Python.

Надає засоби для аналізу коду, графічний відлагоджувач, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA. PyCharm працює під операційними системами Windows, Mac OS X і Linux [20].

Основні можливості:

- Статичний аналіз коду, підсвічування синтаксису і помилок.
- Навігація серед проектів і сирцевого коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використанням методів.

- Рефакторинг: перейменування, витяг методу, введення змінної, введення константи, підняття і опускання методу тощо.
- Інструменти для веб-розробки з використанням фреймворку Django.
- Вбудований відлагоджувач для Python.
- Вбудовані інструменти для юніт-тестування.
- Розробка з використанням Google App Engine.
- Підтримка систем контролю версій: загальний користувацький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття.

3.3 Використання спеціалізованих бібліотек обробки зображень

OpenCV (англ. *Open Source Computer Vision Library*, бібліотека комп'ютерного зору з відкритим кодом) — бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на зображеннях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях [21].

Бібліотека розроблена Intel і нині підтримується Willow Garage та Itseez. Сирцевий код бібліотеки написаний мовою C++ і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua та інших. Може вільно використовуватися в академічних та комерційних цілях.

Numpy — розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. Попередник Numpy, Numeric, був спочатку створений Jim Hugunin. Numpy — відкрите програмне забезпечення і має багато розробників.

Dlib - це загальнопризначена бібліотека програмного забезпечення, написана на мові програмування C ++. Її конструкція в значній мірі залежить від ідей від розробки контракту і розробки програмного забезпечення на основі компонентів. Таким чином, це, перш за все, набір незалежних програмних компонентів. Це програмне забезпечення з відкритим вихідним кодом, що випускається під ліцензією Boost Software [22].

Darknet - це фреймворк з відкритим кодом, написаний на C та CUDA. Він швидкий, простий в установці та підтримує обчислення процесора та GPU. Користувачі можуть знайти джерело на GitHub. Darknet встановлюється лише з двома необов'язковими залежностями: OpenCV, якщо користувачі хочуть широкого спектру підтримуваних типів зображень, або CUDA, якщо вони хочуть обчислення GPU. Це не є обов'язковим, але користувачі можуть почати, просто встановивши базову систему, протестовану лише на комп’ютерах Linux та Mac. У фреймворкці є функція You Look Only Once (YOLO), найсучасніша система виявлення об'єктів у реальному часі. На Titan X він обробляє зображення зі швидкістю 40-90 FPS і має KAP на VOC 2007 78,6% та mAP 44,0% на тестовому розробці COCO. Користувачі можуть використовувати Darknet для класифікації зображень для завдання 1000N & nbsp; ImageNet. Darknet відображає інформацію під час завантаження конфігураційного файлу та ваги, після чого класифікує зображення та друкує топ-10 класів для зображення [23].

3.4 Алгоритм функціонування програми

Додаток було розроблено в вигляді скрипта Python. Схема алгоритму роботи програми зображена на рисунку 3.2.

1) Спочатку здійснюється вибір вхідних параметрів функціонування (1).

2) Варіантами функціонування можуть бути : а)функціонування в реальному часі (3); б) функціонування в режимі обробки(4). В першому

випадку відбувається завантаження веб-камери. В другому — програмам проходить по папці і завантажує медіа файли, які там є.

3) Після цього ми виймаємо кожен фрейм із відео (5), або завантажуємо зображення за допомогою бібліотеки OpenCV. Фрейм переводиться в маств даних і відповідно кожному пікселю на відео буде присвоєно масив із 3 значень, що відповідають кольору пікселя.

4) Масив зображення подається на вхіж мережі розбиваючи зображення на частини. Можна подавати одразу декілька фреймів, для підвищення швидкості обробки, але це потребує обчислювальних апаратних потужностей. Мережа виділяє ознаки (6) і зберігає вектор ознак, що відповідає даним обличчям (7).

5) Фінальним кроком є класифікація зображення (8). Якщо в базі обличчя було відоме, результатом буде прямокутник на зображенні з віділиним обличчям і написом імені.

6) Далі алгоритм повторюється доти, доки будуть ще фрейми зображень(9).

Кожен модуль програми може використовуватись незалежно один від одного. Діаграма класів зображена на рисунку 3.1.

Усі модулі можуть використовуватись на різних серверах, відповідно до того яке в них призначення. Таким чином можна використовувати менеджер повідомлень між серверами.

На рисунку 3.3 зображена структура програми детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж.

Спочатку зображення зчитується за допомогою бібліотеки комп’ютерного зору, перетворюючи дані зображення в багатовимірний масив бібліотеки numpy для подальшої оптимізованої роботи із зображенням.

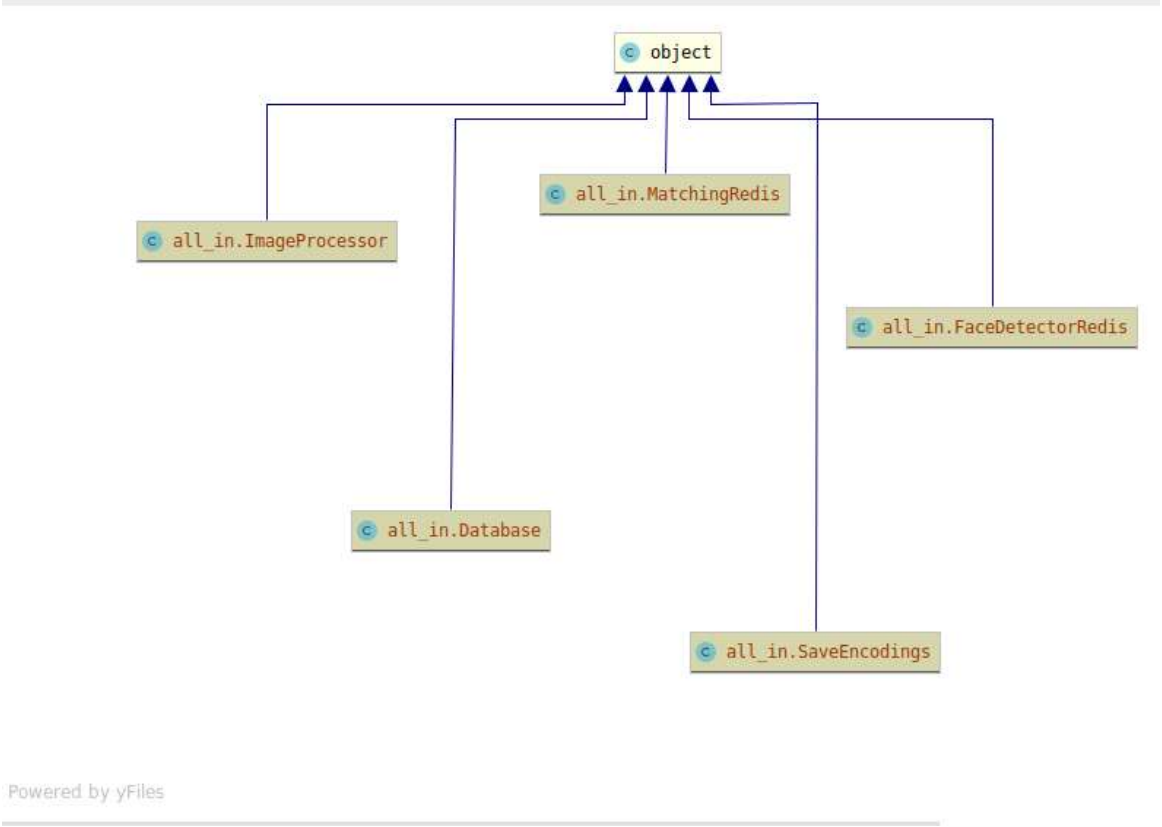

Рисунок 3.1 — Діаграма класів додатку розпізнавання

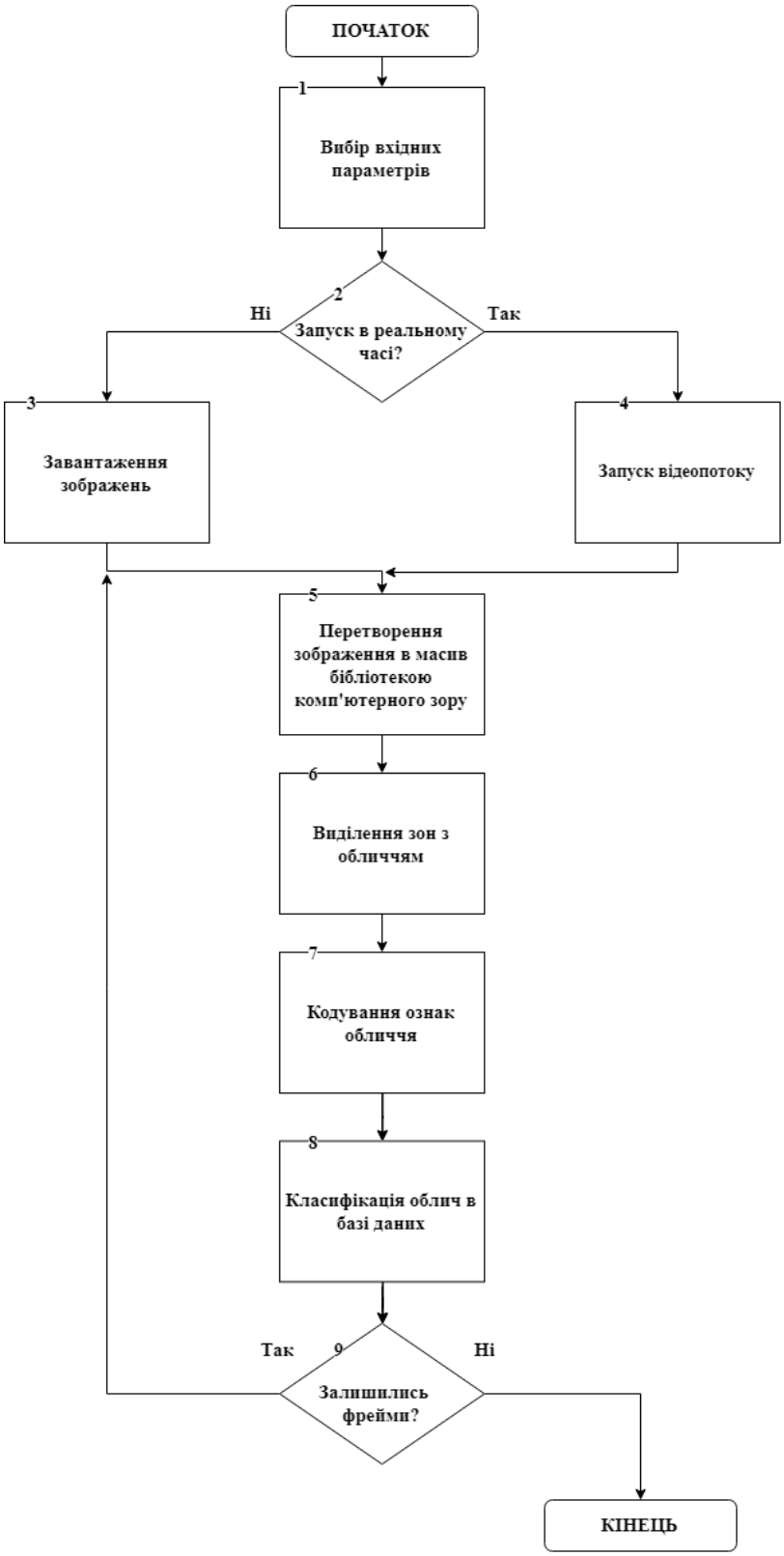

Рисунок 3.2 — Схема загального алгоритму роботи програми

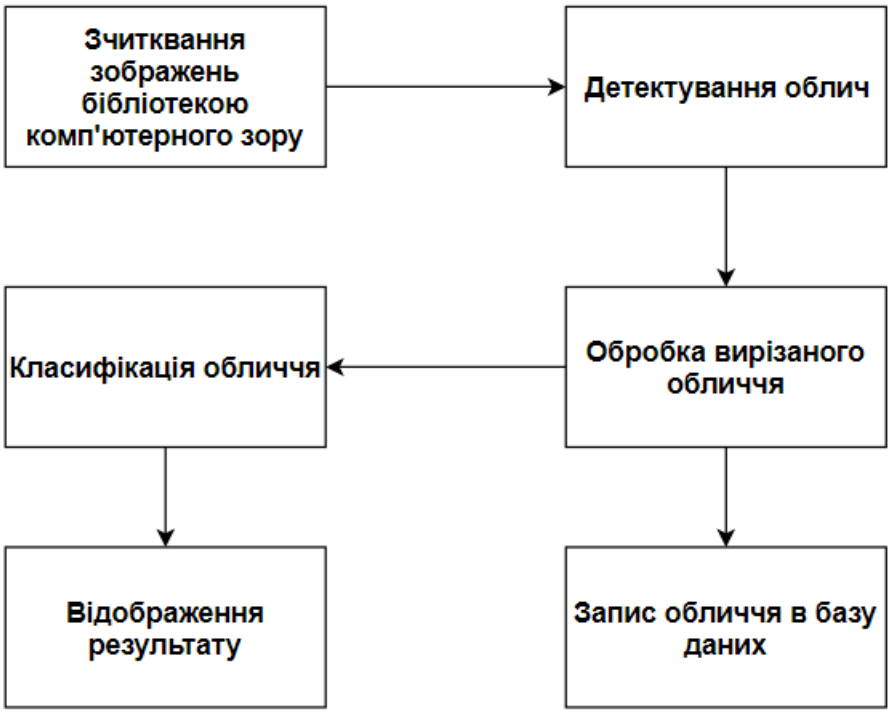

Рисунок 3.3 — Структура програми детектування та розпізнавання

Потім йде детектування після якого, відомі зони з обличчями на зображенні. Після цього ці зони вирізаються із зображення і виконується етап вирівнювання обличчя по основним ознакам обличчя та витягування ознак за, якими буде формуватися вектор для порівняння облич.

Ознаки зберігаються в базу даних і проводиться порівняння вектору ознак і відповідно до відстані векторів з бази даних і обличчя яке оброблюється повертається результат і біля зони обличчя знайденої детектором відображається надпис імені, якому обличчя належить.

3.5 Тестування та аналіз результатів роботи прогами.

Основною метою даної кваліфікаційної роботи було зменшення часу обробки зображення для здійснення детекції. Було проведено навчання дететору, після на наборі даних Labeled Faces in the Wild.

Labeled Faces in the Wild — це база даних зображень обличчя, призначена для вивчення проблеми розпізнавання обличчя в реальному часі. Цю базу даних створили та підтримують дослідники з Університету Массачусетса, штат Амхерст. Детектор обличчя Віоли Джонс виявив і зібрав з Інтернету 13 233 зображення 5 749 людей, із яких 1680 зображених людей мають дві чи більше чітких зображень із загального набору даних. Оригінальна база даних містить чотири різних набори зображень LFW, а також три різні типи "вирівняних" зображень. На думку дослідників, зображення з глибокими каналами, які присутні в даному наборі даних, давали чудові результати для більшості алгоритмів перевірки обличчя порівняно з іншими типами зображень.

Навчання було здійснено за допомогою фреймворку Darknet, який був описаний в минулих розділах. На рисунку 3.3 зображено фінальний графік тренування мережі.

Для того, щоб оцінювати точність детектування зазвичай використовують функцію mAP (mean Average Precision). Вона показує не тільки процент точно розпізнаних зображень, а також враховує, на скільки точно було проведено прогнозування зони детектування з шуканим об’єктом. Враховується пороговий процент зони покриття прогнозу із реальним розміщенням об’єкту.

Для того, щоб оцінювати точність детектування зазвичай використовують функцію mAP (mean Average Precision). Вона показує не тільки процент точно розпізнаних зображень, а також враховує, на скільки точно було проведено прогнозування зони детектування з шуканим об’єктом. Враховується пороговий процент зони покриття прогнозу із реальним розміщенням об’єкту. На рисунку 3.2 видно дану метрику, що склала 99.0 відсотка.

Для нашого навчання було взято, 7% від загальної кількості зображень для оцінки точності. Як видно фінальна точність склала 99.0 відсотків, після 83 тисяч ітерацій. Кожна ітерація містила 64 зображення, які були штучно

згенеровані довільним чином із зображень із основного набору даних, фреймворком для того, щоб збільшити кількість унікальних зображень із наявних. Після цього було виміряно час детектування, тобто час за який мережа після подачі на неї каналів зображення, віддавала результат. Таким чином було взято значення після 100 детектувань на різних зображенням з однаковою роздільною здатністю і з цих значень обраховано середній результат. Результат для звичайної мережі склав 11 мілісекунд. Для вимірювання даного показника була взята функція timeit.

Результат функції показано на рисунку 3.5.

Наступним кроком було здійснення методу відсічення ваг. Після відсічення потрібно тонко налаштувати мережу для того, щоб після відсікання каналів мережі, суміжні ваги, до яких були приєднані, відсічення канали не занулювали, свої значення при проведені прогнозування, а також для того, щоб канали які залишились підігнали перелаштували свої ваги, для збереження. Результати фінального налагодження зображено на рисунку 3.6.

При цьому показник метрики mAP не змінився і залишився 99.0 відсотків. Таким чинок вдалося виконати налаштування без втрати початкової точності.

Фінальне вимірювання часу одного детектування показано на рисунку 3.7 і склало 8 мілісекунд. Вимірювання було здійснено за допомогою функції %timeit. Це метод магії класу. Магічні методи в Python - це спеціальні методи, які додають "магію" вашому класу.

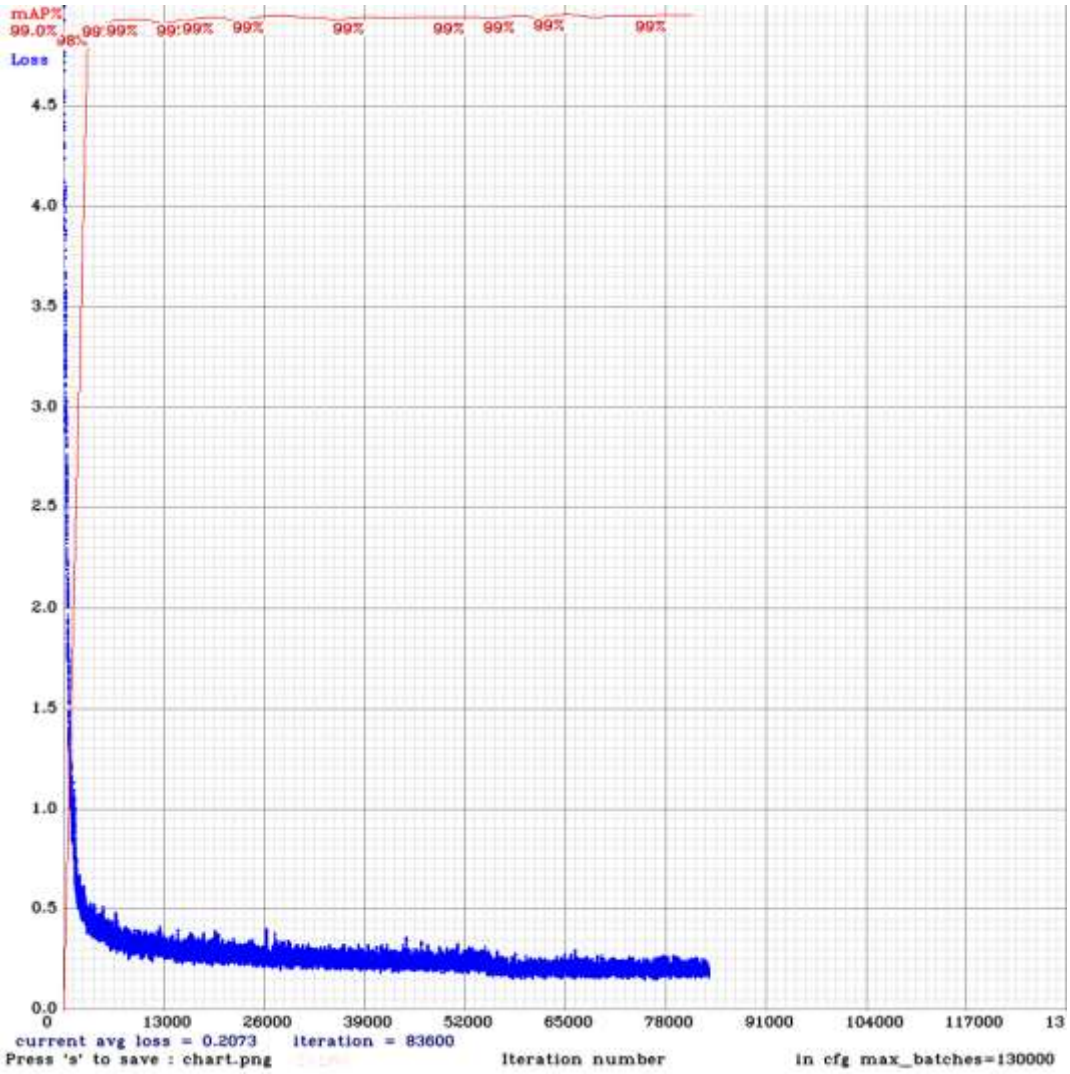

Рисунок 3.4 — Графік повного процесу навчання детектору

```
In [14]: detector.detect(img)

         8.41 ms ± 37.4 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

In [ ]:
```

Рисунок 3.5 — Результат тестування

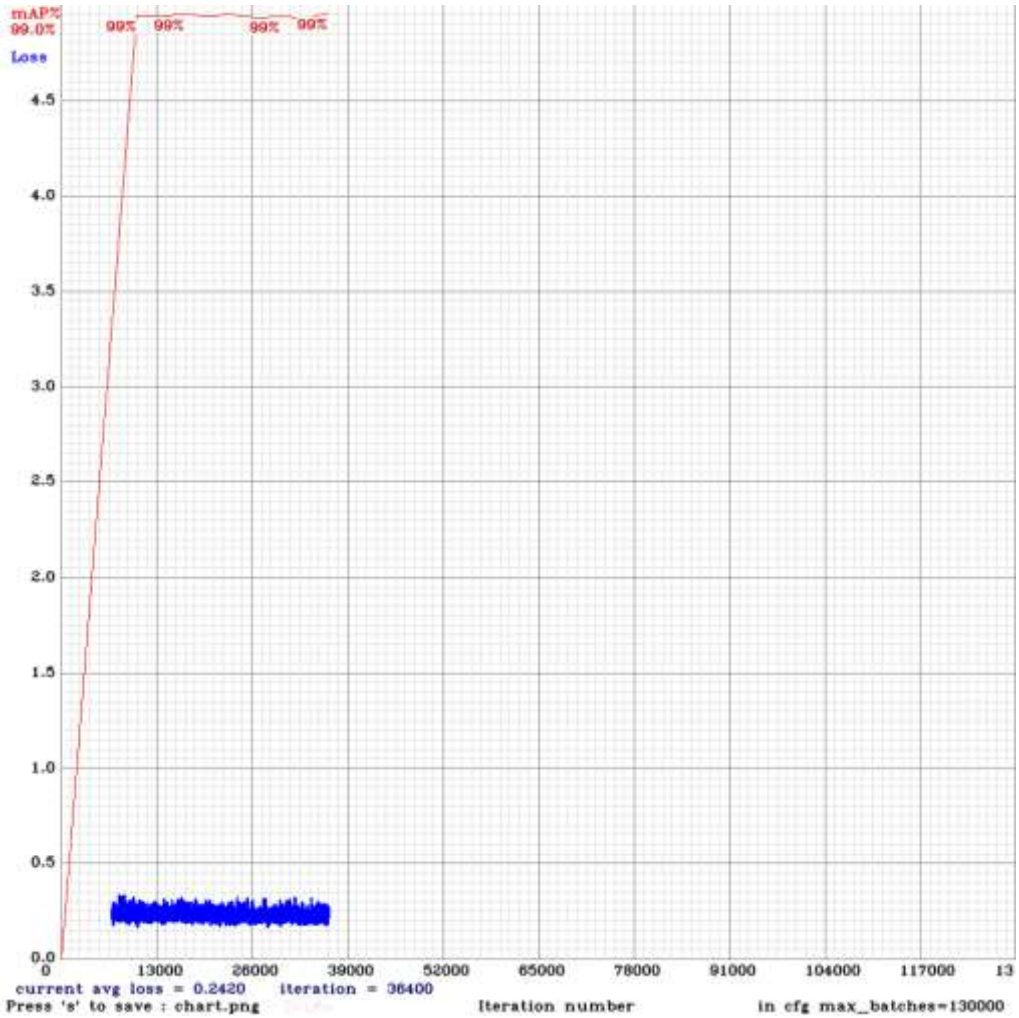

Рисунок 3.6 — Графік тонкого налаштування детектору

Магічні методи не визиваються безпосередньо при виконанні пайтон файлу, їх виклик відбувається всередині з класу під час певної дії. Дана функція аналізує 100 ітерацій і видає точний середній результат і не впливає на час вимірювання. Тестування проводилося над функцією detect(), яка приймає на вхід зображення і вертає координати крайньої лівої верхньої точки, та крайньої нижньої правої точки, що дає змогу виділити зону обличчя.

Як видно з наведених показників вдалося збільшити швидкість детектування на 35 відсотків.

Рисунок 3.7 — Вимірювання часу детектування після налагодження мережі

Результати порівнянь з результатами детектувань програм аналогів зображено в таблиці 3.1. Як бачимо з таблиці тільки деякі з детекторів були швидше початкової мережі до проведення оптимізації. Після збільшення швидкодії детектування мережа показала найкращий результат із усіх програм аналогів приведених в даному порівнянні. Результати порівняння швидкодії і точності наведені в таблиці 3.2.

Останнім кроком є перевірка роботи усієї система, яка складається з компоненту наступних основних компонентів:

— Збір зображень;

— Детектування;

— Обробка зображення;

— Кодування ознак;

— Порівняння з базою кодувань облич.

Фінальним результатом роботи є програма, яка в реальному часі забирає відеопотік з веб-камери, позначає у вікні програми усі обличчя у вигляді рамки і підписує кому, належить обличчя, якщо в базі таке збережено.

Таблиця 3.2 – Порівняння швидкості і точності відомих програм розпізнавання.

Реалізація	Точність	Швидкодія
Innovatrics	98.0%	34мс
Kairos	95.5%	41мс
FaceRecognition	98.38%	23мс
Normal Yolov3 mini	99.0%	11мс
Покращений Yolov3 mini	99.0%	8мс

Вікно програми показано на рисунку 3.8.

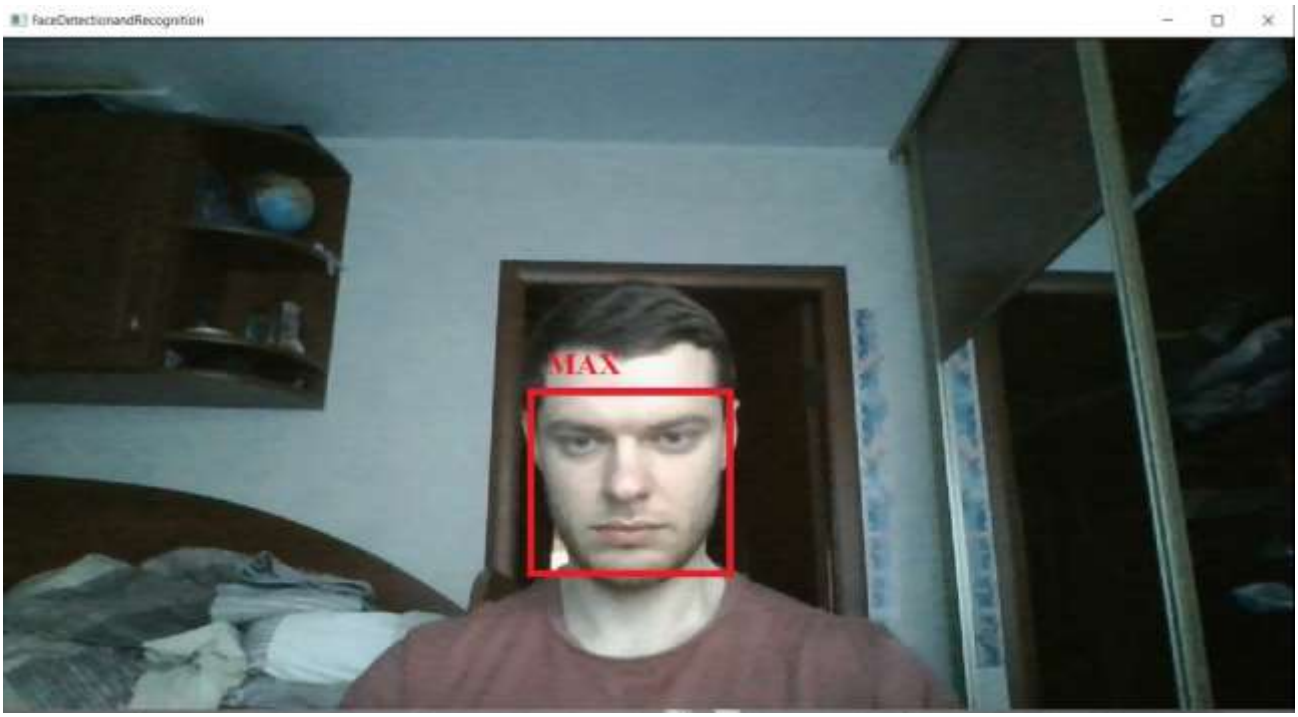

Рисунок 3.8 — Результат роботи програми детектування та розпізнавання

Обличчя обведено червоною рамкою на якою відображається надпис, кому воно належить. Так як швидкість усього процесу розпізнавання займає всього 20 мс процес може здійснюватися в реальному часі на відео.

3.6 Висновок

У даному розділі було розроблено загальну схему алгоритмів функціонування системи детектування та розпізнавання облич на зображеннях. На основі цього було програмно реалізовано систему. Було описано і обґрунтовано вибір інструментів для створення програми.

Було проведено тестування створеної програми, яке підтвердило коректність її роботи. Також було проведено порівняльний аналіз швидкодії детектування з початковим, який використовується в програмах аналогах.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. доцент кафедри КН Колесницький О.К та доцент кафедри КН Арсенюк І.Р.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	3
6	4	4
7	4	3
8	3	4
9	4	3
10	4	4
11	3	3

Продовження таблиці 4.1

12	4	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 42$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р = 21$ день;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 9404,7 = 940,47 \text{ (грн.)}$$

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5000	238,1	8	1904,8
Інженер-програміст	3500	166,66	45	7499,9
Всього:				9404,7

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (785,7+ 7857)\cdot \frac{36,3}{100} = 3755,3 \text{ (грн.)}. \quad (4.2)$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot \frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програм	Балансова вартість,	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний	9000	25	3	562,5
Всього:				562,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n \text{Н}_\text{і} \cdot \text{Ц}_\text{і} \cdot \text{К}_\text{і}, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	150	1	150
Пачка паперу	уп.	100	1	100
Ручка	шт.	5	1	5

Всього з урахуванням транспортних витрат	280,5

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п ; \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,66 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,8$).

$$В_е= 1,66 \cdot 0,6 \cdot 200 \cdot 0,8 = 163,2 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (9404,7 + 940,47) = 10345,17 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 9404,7+940,47+3755,3+562,5+280,5+163,2+10345,17 = 25481,54 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$\text{В}_{\text{заг}} = \frac{\text{В}_{\text{ін}}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$\text{В}_{\text{заг}} = \frac{25481{,}54}{1} = 25481{,}54$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$\text{ЗВ} = \frac{\text{В}_{\text{заг}}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$\text{ЗВ} = \frac{25481{,}54}{0{,}9} = 28279{,}82 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку

підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 30 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 30 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1000 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 30 \cdot 1000 + (400 + 30) \cdot 200 = 116000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 30 \cdot 1000 + (400 + 30) \cdot (200 + 150) = 180500 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 30 \cdot 1000 + (400 + 30) \cdot (200 + 150 + 100) = 223500 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \quad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

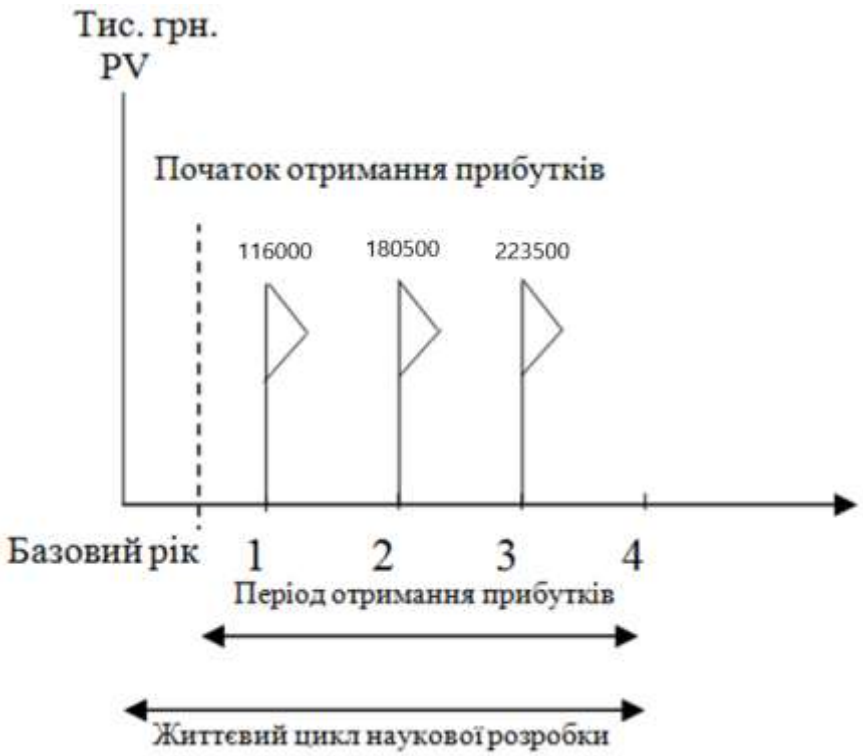

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{28279,82}{(1+0,1)^0} + \frac{116000}{(1+0,1)^2} + \frac{180500}{(1+0,1)^3} + \frac{223500}{(1+0,1)^4} = 412413,4 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 412413,4 - 28279,82 = 384133,58$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (4.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{384133,58}{28279,82}} - 1 = 1,44 \text{ або } 144\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 144\% > \tau_{мін} = 0{,}3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1{,}44} = 0{,}69 \text{ років}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

4.5 Висновок

На основі зроблених підрахунків в економічній частині магістерської кваліфікаційної роботи досягнуті наступні результати:

Встановлено, що рівень комерційного потенціалу розробки є задовільним і доцільним. Для успішної конкурентоспроможності програмного продукту на ринку планується збільшити універсальність програми, а також поповнити функціонал. Таким потрібно зробити рішення, для обчислень на хмарі, з веб інтерфейсом для користувача, щоб забезпечить більший попит у користувачів через збільшення простоти використання і доступність, а також

можлива автоматизація бізнес процесів, або інших процесів, які базуються на відеоспостереженні.

ВИСНОВКИ

У ході виконання магістерської дипломної роботи було реалізовано програму детектування та розпізнавання облич на зображеннях на основі згорткових нейронних мереж. Було проведено аналіз сучасних програм-аналогів, які виконують детектування та розпізнавання та наведено коротку порівняльну характеристику знайдених програм-аналогів. Таким чином було описано проблему і поставлено задачу.

Також було розроблено покращену архітектуру згорткової нейронної мережі для детектування облич, досліджено методи, які можуть бути використані, для задачі розпізнавання. Було запропонувати використати метод відсікання лишніх шарів відомої широко використовуваної згорткової мережі для підвищення швидкодії розпізнавання.

Спроектовано схему алгоритму роботи програми розроблюваної програми, що дає змогу вирішити поставлену задачу. На основі отриманих результатів було створено UML-діаграму класів та . Відповідно до математичної моделі, схеми алгоритму та UML-діаграми класів реалізовано програму. Програмне забезпечення розроблене на мові програмування Python. В результаті роботи програми було встановлено, що розроблювальна програма відповідає вимогам, отже, поставлену задачу вирішено.

Тестування програми пройшло успішно, й довело доцільність розробки даної системи. Порівняно з обраним прототипом швидкодія, при однаковій точності прогнозування, зросла на 35%, що означає досягнення поставленої мети.

Результати виконання магістерської дипломної роботи повністю відповідають технічному завданню.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мазур М.В. Практичне застосування спайкінгових нейронних мереж. Нткп Внту. Факультет інформаційних технологій та комп'ютерної інженерії. / Мазур М. В. [Електронний ресурс] – режим доступу : https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2018/paper/view/4383 (дата звернення 22.10.2019) – Назва з екрана.

2. Мазур М.В. Задача виявлення і класифікації об'єктів за допомогою згорктових нейронних мереж. Нткп Внту. Факультет інформаційних технологій та комп'ютерної інженерії. / Мазур М. В. [Електронний ресурс] – режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2019/paper/view/7698 (дата звернення 22.10.2019) – Назва з екрана.

3. Мазур М.В. Problem of detection and classification of objects with convolutional neural networks. Внту. Молодь в науці: дослідження, проблеми, перспективи (МН-2019). / Кюльян І.Г. [Електронний ресурс] – режим доступу: https://conferences.vntu.edu. ua/index.php/mn/mn2019/paper/view/8157 (дата звернення 22.10.2019) – Назва з екрана.

4. Свідоцтво про реєстрацію авторського права на твір № 89015. Комп’ютерна програма «Моделювання спайкінгового нейрона» / Колесницький О.К., Мазур М.В. Дата реєстрації Державною службою інтелектуальної власності України 29.05.2019.

5. Новотарський М.А. Нестеренко Б.Б. Штучні нейронні обчислення : навчальний посібник / Новотарський М.А. Нестеренко Б.Б. - Інститут математики Національної академії наук України, Київ — 385 с.

6. Руденко О. В. Штучні нейронні мережі : навчальний посібник / О. В. Руденко, Є. В. Бодянський. Харків : ТОВ «Компанія СМІТ», 2006. — 404 с.

7. Колесницький О. К. Методи і засоби розпізнавання сигналів мультисенсорів газів на основі імпульсних нейронних мереж : монографія / Колесницький О. К. — Вінниця : ВНТУ, 2011. — 120 с.

8. Dana H. Ballard; Christopher M. Brown (1982). Computer Vision. Prentice Hall. ISBN 0-13-165316-4.

9. Huang, T. (1996-11-19). Vandoni, Carlo, E, ed. Computer Vision : Evolution And Promise.

10. Milan Sonka; Vaclav Hlavac; Roger Boyle (2008). *Image Processing, Analysis, and Machine Vision*. Thomson.

11. AlexNet [Електронний ресурс] – режим доступу: https://en.wikipedia.org/wiki/AlexNet (дата звернення 22.10.2019) – Назва з екрана.

12. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [Електронний ресурс] – режим доступу: https://arxiv.org/pdf/1506.01497.pdf (дата звернення 22.10.2019) – Назва з екрана.

13. MobileNet version 2 [Електронний ресурс] – режим доступу: https://machinethink.net/blog/mobilenet-v2/ (дата звернення 22.10.2019) – Назва з екрана.

14. YOLO — You only look once, real time object detection explained [Електронний ресурс] – режим доступу: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006 (дата звернення 22.10.2019) – Назва з екрана.

15. SmartFace [Електронний ресурс] – режим доступу: https://www.innovatrics.com/face-recognition-solutions/ (дата звернення 22.10.2019) – Назва з екрана.

16. Recognize People The Way You Want [Електронний ресурс] – режим доступу: https://www.kairos.com/ (дата звернення 22.10.2019) – Назва з екрана.

17. Face Recognition [Електронний ресурс] – режим доступу: https://github.com/ageitgey/face_recognition (дата звернення 22.10.2019) – Назва з екрана.

18. Julien Danjou Serious Python Black - Belt Advice on Deployment , Scalability , testing , and More. / Julien Danjou San Francisco 242c.

19. Peter Farrell math adventures with python an illustrated guide to exploring math with code / Peter Farrell San Francisco 347c.

20. PyCharm [Електронний ресурс] – режим доступу: https://www.jetbrains.com/ru-ru/pycharm/ (дата звернення 22.10.2019) – Назва з екрана.

21. OpenCV [Електронний ресурс] – режим доступу: https://opencv.org/ (дата звернення 22.10.2019) – Назва з екрана.

22. Dlib c++ Library [Електронний ресурс] – режим доступу: http://dlib.net/ (дата звернення 22.10.2019) – Назва з екрана.

23. YOLO: Real-Time Object Detection [Електронний ресурс] – режим доступу: https://pjreddie.com/darknet/yolo/ (дата звернення 22.10.2019) – Назва з екрана.