МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ РОЗПІЗНАВАННЯ ЕМОЦІЙ»**

Виконав: студент 2 курсу, групи 8.1218

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

І.О. Третяк
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної математики, доцент, к.ф.-м.н. Ткаченко І.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, доцент, к.ф.-м.н.

_____ Лісняк А.О.
(підпис)

« 29 » травня 2019 р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Третяку Ігору Олеговичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка системи комп'ютерного зору для розпізнавання емоцій

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи 26 грудня 2019 року

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі
2. Основні теоретичні відомості

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.05.2019

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	05.09.2019	
2.	Збір вихідних даних.	13.09.2019	
3.	Обробка методичних та теоретичних	29.09.2019	
	джерел.		
4.	Розробка першого та другого розділу.	10.10.2019	
5.	Розробка третього розділу.	23.11.2019	
6.	Оформлення та нормоконтроль	27.12.2019	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	15.01.2020	

Студент _____ І.О. Третяк
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка системи комп'ютерного зору для розпізнавання емоцій»: 52 сторінки, 15 рисунків, 1 таблиця, 8 джерел, 1 додаток.

ГЛИБОКЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ЕМОЦІЙ, KERAS, OPENCV, TENSORFLOW.

Об’єкт дослідження – сучасні системи комп'ютерного зору.

Мета роботи: розробка системи комп’ютерного зору для розпізнавання емоцій.

Метод дослідження – аналітичний.

У роботі досліджуються сучасні системи комп'ютерного зору. Розглядаються різні засоби та технології для вирішення проблеми виявлення та класифікації об’єкту на зображенні, запропоноване рішення на основі якого розробляється дизайн системи.

В роботі представлен детальний огляд технологій і засобів для виявлення та класифікації об’єктів на зображенні.

SUMMARY

Master’s Qualification Thesis «Development of a Computer Vision System for Facial Expression Recognition»: 52 pages, 15 images, 1 table, 8 sources, 1 supplement. COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORK, DEEP LEARNING, EMOTION RECOGNITION, KERAS, OPENCV, TENSORFLOW.

The object of the study is modern computer vision systems.

The aim of the study is design of computer vision system for emotion recognation.

The method of research is analytical.

In the work modern systems of computer vision are investigated. Various tools and technologies are being considered to solve the problem of detecting and classifying an object in an image, and a solution is proposed to design the system.

The paper provides a detailed overview of technologies and tools for identifying and classifying objects in an image.

ЗМІСТ

ВСТУП

Перші комп’ютери були створені виключно для математичних обчислень, а вже менше ніж через століття у кожного у домі декілька пристроїв, які можуть виконувати мільйони операцій в секунду. За цей невеликий період комп’ютерні системи проникли в усі аспекти людського життя.

На сьогоднішній день за допомогою комп’ютерних систем автоматизується широкий спектр задач та процесів, які в недалекому минулому покладалися лише на людей. Комп’ютерні системи використовуються повсюди: в освіті, в промисловості, в медицині тощо.

Зараз нас повсюди оточують пристрої, які генерують терабайти зображень та відеоданих. Наприклад смартфони, камери спостереження, різноманітні датчики в офісних приміщеннях та на підприємствах. Але до недавнього часу проаналізувати їх могла лише людина.

Галузь, яка займається розробкою систем, здатних аналізувати візуальні дані, називають комп’ютерним зором. Вони отримують інформацію у вигляді зображень. Відеодані можуть бути представлені у вигляді багатьох форм, таких як відеопослідовність, зображення з різних камер або тривимірними даними з медичного сканера.

Розвиток нейронних мереж і технології глибокого навчання, полегшили вирішення задач комп’ютерного зору, такі як класифікація зображень, виявлення об’єктів на зображенні, відновлення зображення тощо.

Комп’ютерний зір вже застосовується для вирішення багатьох важливих проблем у різних сферах людської діяльності: в медицині, для виявлення пухлин, атеросклерозу чи інших злоякісних змін; в промисловості, для автоматичної перевірки кінцевого продукту на наявність дефектів; у військовій галузі, для виявлення ворожих солдат і транспортних засобів та управління ракетами, а також для створення автономних транспортних засобів.

Метою роботи, являється огляд сучасних систем комп’ютерного зору та створення система комп’ютерного зору для класифікації емоцій. Цю систему можна використовувати як допоміжний інструмент в психологічних онлайн консультаціях, або в сфері послуг, для того щоб фіксувати наскільки клієнт задоволений роботою персоналу.

1 ОГЛЯД ТЕХНОЛОГІЙ

1.1 Теорія розпізнавання образів

Теорія розпізнавання образів являє собою розділ інформатики та суміжних дисциплін, що розвиває методи класифікації та ідентифікації об’єктів різної природи: сигналів, ситуацій, предметів, що характеризуються кінцевою множиною деяких ознак.

Проблема розпізнавання образів також входить в поле міждисциплінарних досліджень – в тому числі в зв’язку з роботою зі створення штучного інтелекту, а також часто використовується при вирішенні практичних завдань галузі комп’ютерного зору, до чого відноситься і розглянутий випадок.

При постановці класичної задачі розпізнавання образів прийнято використовувати строгу математичну мову, ґрунтуючись на логічних міркуваннях і математичних доказах. На противагу цьому підходу, існують методи розпізнавання образів з використанням машинного навчання і штучних нейронних мереж, сформовані не настільки строго формалізованими підходами до розпізнавання, але, як буде показано далі, демонструє не гірший, а в деяких випадках значно перевершує класичні методи результат.

Характер проблематики розпізнавання образів також обмежує сферу застосування різних традиційних методів вузькими спеціалізованими напрямками, в кожному з яких найбільш ефективними виявляються одні методи і неефективними інші. Цей фактор обумовлює складність спільного рішення проблеми індукції і в цій дисципліні.

1.2 Комп’ютерний зір

Комп’ютерним зором називають галузь інформатики, яка фокусується на реплікації частин системи зору людини та надає можливість комп’ютерам ідентифікувати та обробляти об’єкти в зображеннях та відео так, як це роблять люди.

КЗ також є науковою дисципліною, що базується на теорії створення та використання моделей та систем, які призначені для виділення інформації з зображень. Основою для дослідження є не лише статичні зображення, але й відеодані (послідовність зображень, зображення з камер, тривимірні дані тощо).

Як технологічна дисципліна, КЗ прагне застосувати теорії та моделі КЗ до створення систем КЗ. Прикладами таких систем можуть бути:

- системи керування процесами (промислові роботи, автономні транспортні засоби, машинний зір);
- системи відеоспостереження;
- системи організації інформації (наприклад, для індексації баз даних зображень);
- системи моделювання об’єктів або оточуючого середовища (в медицині: аналіз медичних зображень; у військовій справі: аналіз топографічних даних, системи виявлення ворожих військових сил; системи виявлення лісових пожеж тощо);
- системи взаємодії (наприклад, пристрої введення для систем людино-машинної взаємодії);
- системи доповненої реальності;
- обчислювальна фотографія (наприклад для мобільних пристроїв з камерами).

Варто відзначити, що дана область можна охарактеризувати як досить молоду та швидко розвиваючуюся. І, хоча існують більш ранні роботи, можна сказати, що тільки з кінця 1970-х почалось інтенсивне вивчення цієї проблеми, коли комп’ютери змогли керувати обробкою великих наборів даних, таких як

зображення. Однак, ці дослідження зазвичай починались з інших галузей, і, відповідно, нема стандартного формулювання проблеми КЗ. Також, і це навіть більш важливо, нема стандартного формулювання того, як повинна вирішуватись проблема КЗ. Замість того, існує маса методів для вирішення різноманітних строго визначених задач КЗ, де методи часто залежать від задач і рідко коли можуть бути узагальнені для широкого кола застосування. Багато з методів та застосувань все ще знаходяться на стадії фундаментальних досліджень, але все більша кількість методів знаходить застосування в комерційних продуктах, де вони часто складають частину складнішої системи

Завдяки прогресу в галузі штучного інтелекту та інновацій у глибокому навчанні та нейронних мережах, в останні роки системи КЗ змогли зробити значний стрибок і навіть перевершити людей у деяких завданнях, пов’язаних з виявленням та маркуванням об’єктів.

Одним з рушійних факторів активного росту галузі КЗ є кількість даних, які ми сьогодні генеруємо, і які потім використовуються для навчання та покращення КЗ.

Обчислювальна потужність, необхідна для аналізу даних, стала доступнішою. Оскільки галузь КЗ розвивається за допомогою нового апаратного забезпечення та нових алгоритмів, так само зростає і рівень точності ідентифікації об’єктів. Менш ніж за десятиліття сьогоднішні системи досягли майже 99% точності у порівнянні з 50%.

1.3 Основні задачі комп’ютерного зору

1.3.1 Класифікація об’єкту

Класифікація та локалізація являються двома основними задачами КЗ. Класифікація зображень в основному передбачає лише маркування зображення на основі змісту зображення. Зазвичай існує фіксований набір міток, і ваша

модель повинна буде передбачити мітку, яка найкраще відповідає зображенню. Ця задача, безумовно, важка для машини, оскільки все, що вона бачить, – це лише потік чисел на зображенні.

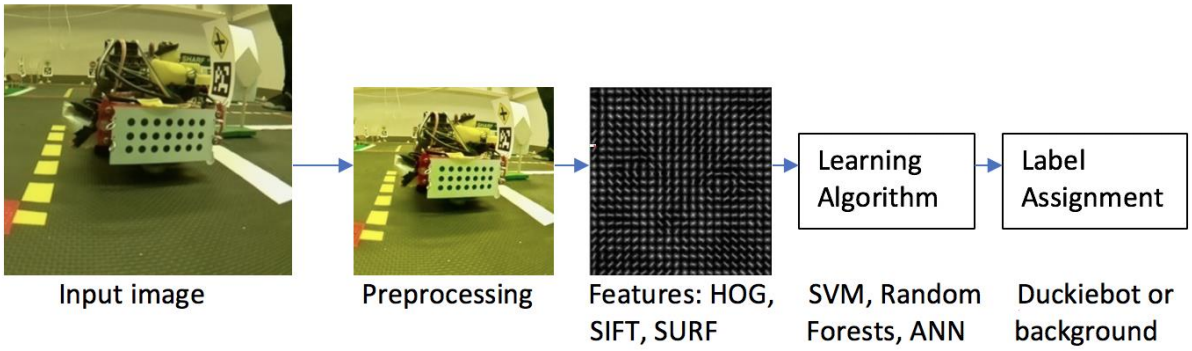

Рисунок 1.1 – Процес класифікації об’єктів

Традиційні способи класифікації об’єктів вилучають особливості або дескриптори для різних класів, а потім використовують класифікатори для класифікації об’єктів. Типовими дескрипторами ознак, що використовуються, є гістограма направлених градієнтів (HOG), масштабонезалежне перетворення ознак (SIFT), Хаар-подібні ознаки, стійкі прискорені ознаки (SURF) тощо. На сьогодні, з розвитком ЗНМ, точність класифікації об’єктів надзвичайно покращується .

Одним з практичних прикладів є класифікація дорожніх знаків, що дуже корисно для системи автопілоту автомобіля.

1.3.2 Локалізація об’єкту

Це свого роду проміжне завдання між двома іншими завданнями, класифікацією зображень та виявленням об’єктів. У класифікації зображень ви повинні призначити одну мітку зображенню, що відповідає основному об’єкту (зрештою, зображення може містити кілька об’єктів). Локалізація являється

процесом визначення положення одного екземпляра цього об’єкта, навіть якщо зображення містить кілька його екземплярів.

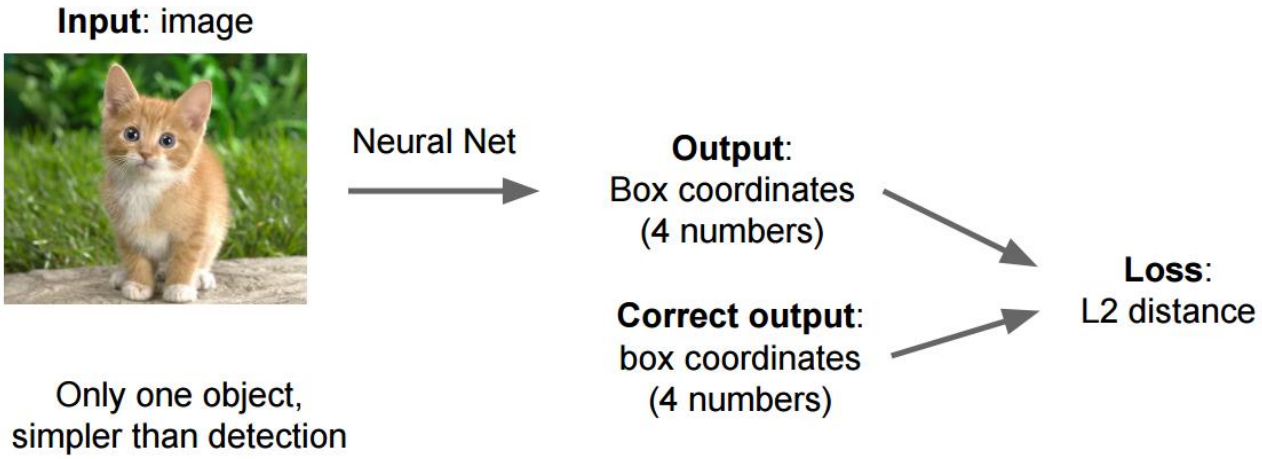

Рисунок 1.2 – Процес навчання системи локалізації об'єктів.

Процес локалізації повинен повернути нам чотири числа: початкові координати (x0, y0), висоту та ширину обмежувальної рамки навколо об’єкту. Потрібно тренувати цю систему використовуючи зображення с завідомо дійсною обмежувальною рамкою та використовуєте відстань L2 для обчислення втрат між передбачуваним обмежувальним полем і істиною.

1.3.3 Виявлення об’єктів

Виявлення об’єктів являється процесом пошуку екземплярів реальних предметів, таких як обличчя, велосипеди та споруди у зображеннях чи відео. Алгоритми виявлення об’єктів зазвичай використовують виділення ознак та алгоритми навчання для розпізнавання екземплярів категорії об’єктів.

Визначення, що надається для виявлення об’єктів ILSVRC 2016, включає вихідні обмежувальні рамки та мітки для окремих об’єктів. Це відрізняється від завдання класифікації та локалізації тим, що застосовує класифікацію та локалізацію до багатьох об'єктів, а не лише одного домінуючого об’єкта.

На сьогоднішній день існує два основні типи алгоритмів для виявлення об’єктів:

– алгоритми, засновані на класифікації. Вони працюють у два етапи: на першому кроці ми вибираємо із зображення цікаві регіони; тоді ми класифікуємо ці регіони за допомогою згорткових нейронних мереж.

Це рішення може бути дуже повільним, оскільки в цьому випадку потрібно прогнозувати клас для кожного вибраного регіону. Найбільш відомим прикладом цього типу алгоритмів є Region-based convolutional neural network (RCNN);

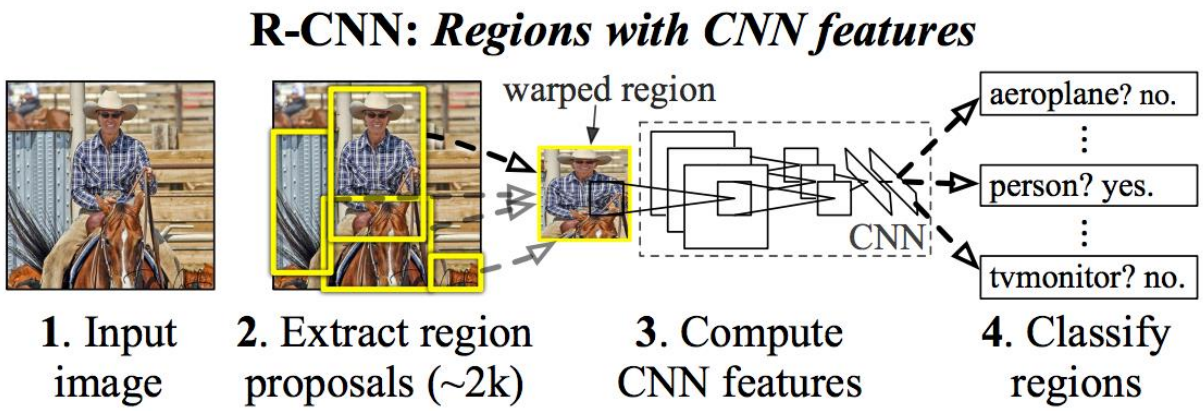

Рисунок 1.3 – Процес виявлення об'єктів використовуючи RCNN

– алгоритми на основі регресії. Замість того, щоб вибирати цікаві частини зображення, ми передбачуємо класи та обмежувальні поля для всього зображення в одному запуску алгоритму. Для цього треба накласти на зображення сітку, розділяючи його на комірки. Кожна комірка намагається передбачити координати зони виявлення з оцінкою впевненості для цих полів і ймовірністю класів. Потім оцінка впевненості для кожної зони виявлення множиться на ймовірність класу, щоб отримати остаточну оцінку.

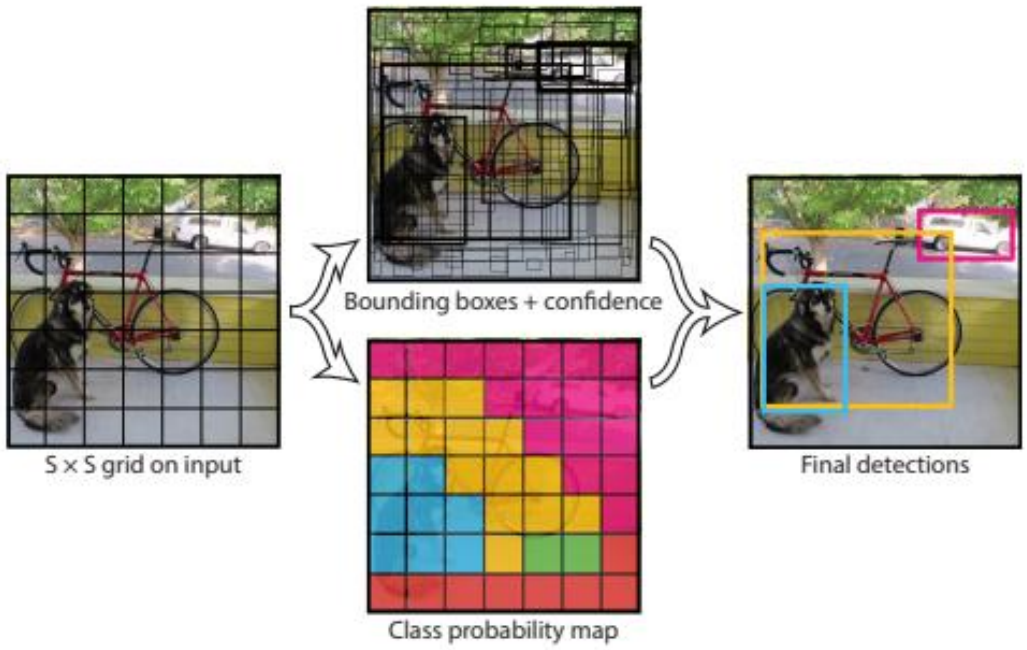

Рисунок 1.4 – Процес виявлення об'єктів використовуючи YOLO

Найбільш відомий приклад цього типу алгоритмів - YOLO (You Only Look Once), що зазвичай використовується для виявлення об'єктів у режимі реального часу.

1.3.4 Сегментація зображення

Сегментація є одним з найбільш унікальних завдань КЗ через те, що мережам потрібно вивчати як інформацію низького, так і високого рівня.

Сегментація являється процесом розділення цифрового зображення на декілька сегментів (множина пікселів, які часто називають суперпікселями). Мета сегментації полягає у спрощенні і зміні представлення зображення для полегшення його аналізу або передачі каналами зв’язку. Сегментацію зображень зазвичай використовують для виділення об’єктів та меж (лінії, криві, і т. д.) на зображеннях. Точніше, сегментація зображень – це процес присвоєння таких міток кожному пікселю зображення, що пікселі з однаковими мітками мають спільні візуальні характеристики.

Результатом сегментації зображення є множина сегментів, які разом покривають все зображення, або множина контурів, виділених з зображення. Всі пікселі в сегменті схожі за деякою характеристикою або за визначеною властивістю, наприклад колір, яскравість або текстура. Сусідні сегменти істотно відрізняються за цими характеристиками.

1.4 Методи покращення зображення

В задачі КЗ часто використовується фільтрація для попередньої обробки зображення перед аналізом його внутрішніх морфологічних ознак.

1.4.1 Бінарізація зображення

Для RGB зображення та зображення в градаціях сірого порогом є значення кольору. Вибір порогу, за яким відбувається бінарізація, визначає процес бінарізації. Зазвичай бінаризація виконується за допомогою адаптивного алгоритму, який визначає поріг. Таким алгоритмом може являтись вибір математичного очікування, моди або піків гістограми. При роботі з гістограмою бінаризація ефективна для сегментації кольорів.

1.4.2 Розмивання зображення

Розмивання зображення по Гаусу шляхом попіксельного віднімання одного гаусіана вихідного зображення від гаусіана з іншим радіусом розмиття. Цей алгоритм широко застосовується в галузі машинного зору і працює достатньо швидко, оскільки існують швидкі способи виконання розмивання за Гаусом. Найважливіші параметри фільтра – два радіуси. Їх найлегше вказати дивлячись на вікно перегляду. Варто пам’ятати, що збільшення малого радіусу

призводить до розширення границь, а зменшення великого радіусу збільшує поріг, за яким визначається межа це чи ні. У більшості випадків кращі результати виходять коли другий радіус менший, ніж перший. Для світлих зображень з темним тлом зворотні значення дають кращі результати.

1.4.3 Виділення границь

Фільтри контурів дуже корисні при вирішенні задач КЗ в разі якщо необхідна реалізація обробки складних об’єктів на зображенні. Фільтрацію контурів реалізує ряд алгоритмів, таких як: оператор Кенні, який є найбільш часто використовуваним алгоритмом пошуку контурів, оператор Собеля, оператор Лапласа, оператор Прюітт, оператор Робертса. Алгоритм Кенні реалізує поняття подавлення немаксимумів, це означає, що пікселями границь оголошуються пікселі, в яких досягається локальний максимум градієнта в напрямку вектора градієнта. В КЗ і обробці зображень при пошуку меж об’єктів крім особливих окремих випадків важко знайти детектор, який би працював істотно краще, ніж детектор Кенні.

1.5 Глибоке навчання в галузі комп’ютерного зору

До появи глибокого навчання, завдання, які комп’ютерне бачення могло виконувати, були дуже обмеженими і вимагали багато ручного кодування та зусиль розробників та операторів. Наприклад, якщо ви хочете виконати розпізнавання обличчя, вам доведеться виконати наступні дії:

- Створення бази даних: доведеться фіксувати окремі зображення всіх предметів, які ви хочете відслідковувати, у певному форматі.
- Анотування зображень: для кожного окремого зображення потрібно буде ввести кілька ключових точок даних, таких як відстань між очима, ширина мосту носа, відстань між верхньою губою та носом та десятки

інших вимірювань, які визначають унікальні характеристики кожної людини.

– Збір нових зображень: доведеться робити нові зображення, будь то фотографії чи відео. І тоді доведеться знову пройти процес вимірювання, позначивши ключові точки на зображенні. Також довелося б враховувати ракурс зображення.

Після всієї цієї ручної роботи додаток, нарешті, зможе порівняти вимірювання на новому зображенні з тими, що зберігаються в його базі даних, і повідомити, чи відповідає він будь-якому з профілів, які він відстежує. Насправді, автоматизації було задіяно дуже мало, і більша частина роботи проводилася вручну. І похибка була ще великою.

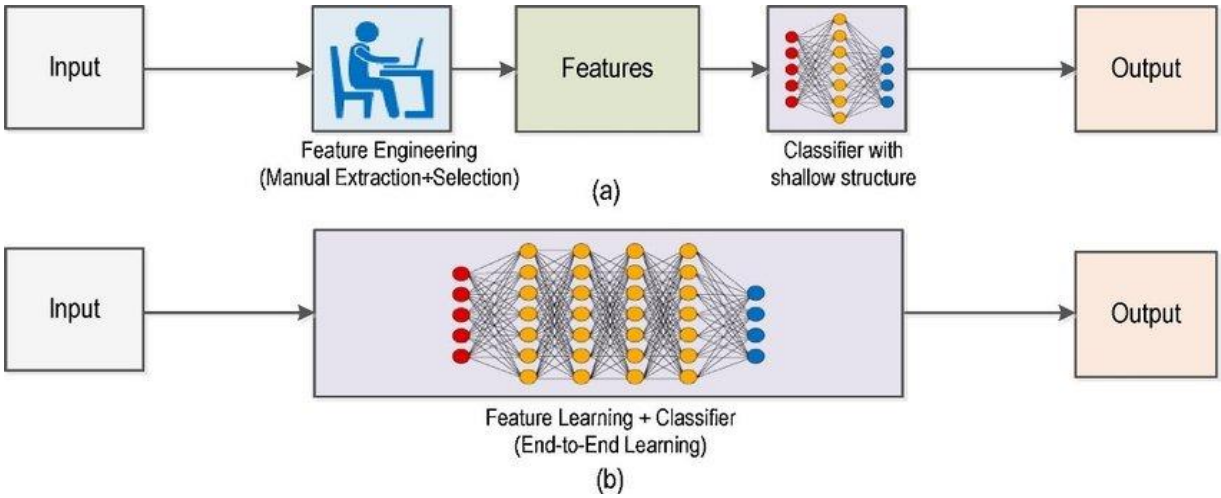

Рисунок 1.5 – Робочий процес комп'ютерного зору без використання глибокого навчання (а) та з використанням глибокого навчання

Машинне навчання передбачало інший підхід до вирішення проблем із КЗ. За допомогою машинного навчання розробникам більше не потрібно вручну кодувати кожне правило в свої програми КЗ. Натомість вони запрограмували менші програми, які могли б виявити патерни зображення. Потім використовували статистичний алгоритм навчання, такий як лінійна регресія, логістична регресія, дерева рішень або метод опорних векторів для виявлення шаблонів та класифікації зображень та виявлення в них об’єктів.

Машинне навчання допомогло вирішити багато проблем, які були історично складними для класичних інструментів та підходів до розробки програмного забезпечення. Наприклад, роки тому інженери машинного навчання змогли створити програмне забезпечення для діагностики раку молочної залози, яке помітно перевершує людину по чутливості, не кажучи вже про швидкість роботи. Однак розробка особливостей програмного забезпечення вимагала зусиль десятків інженерів та експертів з раку молочної залози та зайняла багато часу.

Глибоке навчання дало принципово інший підхід до машинного навчання. ГН покладається на нейронні мережі, функцію загального призначення, яка може вирішити будь-яку проблему, представлену на прикладах. Якщо надати нейронній мережі безліч маркованих прикладів конкретного виду даних, можна отримати спільні патерни між цими прикладами та перетворити їх у математичне рівняння, яке допоможе класифікувати майбутні фрагменти інформації.

Наприклад, створення програми розпізнавання обличчя з поглибленим навчанням вимагає лише розробити або вибрати заздалегідь побудований алгоритм і навчити його з прикладами обличь людей, яких він повинен виявити. Надаючи достатньо прикладів (безліч прикладів), нейронна мережа зможе виявити обличчя без подальших інструкцій щодо особливостей чи вимірювань.

ГН являється дуже ефективним методом КЗ. У більшості випадків створення хорошого алгоритму глибокого навчання зводиться до збору великої кількості маркованих навчальних даних та налаштування таких параметрів, як тип та кількість шарів нейронних мереж та навчальних епох. Порівняно з попередніми типами машинного навчання, ГН є простішим та швидшим для розробки та розгортання.

Більшість сучасних програм КЗ, таких як система діагностики раку, автоматичне керування для автомобіля та розпізнавання обличчя, використовують ГН. ГН та глибокі нейронні мережі перейшли від концептуальної сфери до практичних застосувань завдяки наявності та удосконаленню ресурсів апаратних та хмарних обчислень.

1.6 Згорткова нейронна мережа

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз’єднаних шарів та шарів нормалізації.

Цей процес описують в нейронних мережах як згортку за домовленістю. З математичної точки зору він є радше взаємною кореляцією, ніж згорткою. Це має значення лише для індексів у матриці, й відтак які ваги на якому індексі розташовуються.

1.6.1 Згорткові шари

Згортковий шар є основним будівельним блоком ЗНМ. Параметри шару складаються з набору фільтрів для навчання, які мають невеличке рецептивне поле, але простягаються на всю глибину вхідної ємності. Протягом прямого проходу кожен фільтр здійснює згортку за шириною та висотою вхідної ємності, обчислюючи скалярний добуток даних фільтру та входу, і формуючи 2-вимірну карту збудження цього фільтру. В результаті мережа навчається, які фільтри активуються, коли вона виявляє певний конкретний тип ознаки у певному просторовому положенні у вході.

Складання карт збудження всіх фільтрів уздовж виміру глибини формує повну ємність виходу згорткового шару. Таким чином, кожен запис в ємності виходу може також трактуватися як вихід нейрону, що дивиться на невеличку область у вході, та має спільні параметри з нейронами тієї ж карти збудження.

При опрацюванні входів високої розмірності, таких як зображення, недоцільно з’єднувати нейрони з усіма нейронами попередньої ємності, оскільки така архітектура мережі не бере до уваги просторову структуру даних. Згорткові мережі використовують просторово локальну кореляцію шляхом забезпечення схеми локальної з’єднаності між нейронами сусідніх шарів: кожен нейрон

з’єднано лише з невеликою областю вхідної ємності. Обшир цієї з’єднаності є гіперпараметром, що називається рецептивним полем нейрону. З’єднання є локальними в просторі (вздовж ширини та висоти), але завжди поширюються вздовж усієї глибини вхідної ємності. Така архітектура забезпечує, щоби навчені фільтри виробляли найсильніший відгук до просторово локальних вхідних образів.

Розмір ємності виходу згорткового шару контролюють три гіперпараметри:

- глибина ємності виходу контролює кількість нейронів шару, що з’єднуються з однією й тією ж областю вхідної ємності. Ці нейрони вчаться активуватися для різних ознак входу. Наприклад, якщо перший згортковий шар бере як вхід сире зображення, то різні нейрони вздовж виміру глибини можуть активуватися в присутності різних орієнтованих контурів, або плям кольору;
- крок контролює те, як стовпчики глибини розподіляються за просторовими вимірами (шириною та висотою). Коли кроком є 1, ми рухаємо фільтри на один піксель за раз. Це веде до сильного перекриття рецептивних полів між стовпчиками, а також до великих ємностей виходу. Коли ми робимо крок 2 (або, рідше, 3 чи більше), то фільтри, просуваючись, перестрибують на 2 пікселі за раз. Рецептивні поля перекриваються менше, й отримувана в результаті ємність виходу має менші просторові розміри;
- нульове доповнення. Розмір цього доповнення є третім гіперпараметром. Доповнення забезпечує контроль над просторовим розміром ємності виходу. Зокрема, іноді бажано точно зберігати просторовий розмір вхідної ємності.

Схема спільного використання параметрів застосовується в згорткових шарах для регулювання кількості вільних параметрів. Вона спирається на одне розумне припущення: якщо клаптикова ознака є корисною для обчислення в певному просторовому положенні, то вона також повинна бути корисною для

обчислення й в інших положеннях. Іншими словами, позначаючи 2-вимірний зріз за глибиною як зріз глибини, ми обмежуємо нейрони в кожному зрізі глибини використанням одних і тих же ваг та упередженості.

Оскільки всі нейрони в одному зрізі поділяють спільну параметризацію, то прямий прохід у кожному зрізі глибини згорткового шару може бути обчислено як згортку ваг нейронів із вхідною ємністю (звідси й назва: згортковий шар). Таким чином, є звичним називати набори ваг фільтром (або ядром), який згортається із входом. Результатом цієї згортки є карта збудження, і набір карт збудження для кожного з різних фільтрів складають докупи вздовж виміру глибини для отримання ємності виходу. Спільне використання параметрів сприяє інваріантності архітектури ЗНМ відносно зсуву.

Іноді спільне використання параметрів може й не мати сенсу. Особливо в тому разі, коди вхідні зображення до ЗНМ мають певну особливу центровану структуру, в якій ми очікуємо зовсім різних ознак для навчання в різних просторових положеннях. Одним із практичних прикладів є коли вхід є обличчями, що було відцентровано в зображенні: ми можемо очікувати, що вчитимемося різних особливих ознак очей та волосся в різних частинах зображення. В такому разі є звичним пом’якшувати схему спільного використання параметрів, і натомість просто називати шар локально з’єднаним.

1.6.2 Агрегувальні шари

Іншим важливим поняттям ЗНМ є агрегування, яке є різновидом нелінійного зниження дискретизації. Існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є максимізаційне агрегування. Воно розділяє вхідне зображення на набір прямокутників без перекриттів, і для кожної такої підобласті виводить її максимум. Ідея полягає в тому, що точне положення ознаки не так важливе, як її грубе положення відносно інших ознак. Агрегувальний шар слугує поступовому скороченню просторового розміру представлення для зменшення кількості параметрів та об’єму обчислень

у мережі, і відтак також для контролю перенавчання. В архітектурі ЗНМ є звичним періодично вставляти агрегувальний шар між послідовними згортковими шарами. Операція агрегування забезпечує ще один різновид інваріантності відносно паралельного перенесення.

Агрегувальний шар діє незалежно на кожен зріз глибини входу, і зменшує його просторовий розмір. Найпоширенішим видом є агрегувальний шар із фільтрами розміру 2×2, що застосовуються з кроком 2, який знижує дискретизацію кожного зрізу глибини входу в 2 рази як за шириною, так і за висотою, відкидаючи 75% збуджень. В цьому випадку кожна операція взяття максимуму діє над 4 числами. Розмір за глибиною залишається незмінним.

На додачу до максимізаційного агрегування, агрегувальні вузли можуть використовувати й інші функції, такі як усереднювальне агрегування та L2-нормове агрегування. Історично усереднювальне агрегування застосовувалася часто, але останнім часом впало в немилість у порівнянні з дією максимізаційного агрегування, робота якого на практиці виявилася кращою.

Через агресивне скорочення розміру представлення, тенденція йде до менших фільтрів, або відмови від агрегувального шару взагалі.

Агрегування областей інтересу — це варіація максимізаційного агрегування, в якій розмір виходу фіксовано, а прямокутник входу є параметром.

Агрегування є важливою складовою згорткових нейронних мереж для виявлення об’єктів, що ґрунтуються на архітектурі швидких згорткових нейронних мереж на основі областей.

1.6.3 Шар зрізаних лінійних вузлів

Цей шар застосовує ненасичувальну передавальну функцію f(x)=max(0,x). Він посилює нелінійні властивості функції ухвалення рішення і мережі в цілому, не зачіпаючи рецептивних полів згорткового шару.

Для посилення нелінійності застосовуються й інші функції, наприклад, насичувальні гіперболічний тангенс f(x)=\tanh(x), f(x)=|tanh(x)|, та сигмоїдна

функція f(x)=(1+e^(-x))^(-1). Зрізаному лінійному вузлові часто віддають перевагу перед іншими функціями, оскільки він тренує нейронну мережу в декілька разів швидше без значної розплати точністю узагальнення.

1.6.4 Повноз'єднаний шар

Насамкінець, після кількох згорткових та максимізаційно агрегувальних шарів, високорівневі міркування в нейронній мережі здійснюються повноз'єднаними шарами. Нейрони у повноз'єднаному шарі мають з'єднання з усіма збудженнями попереднього шару, як це можна бачити у звичайних нейронних мережах. Їхні збудження відтак може бути обчислювано матричним множенням, за яким слідує зсув упередженості.

1.6.5 Шар втрат

Шар втрат визначає, як тренування штрафує відхилення між передбаченими та справжніми мітками, і є, як правило, завершальним шаром. Для різних завдань у ньому можуть використовувати різні функції втрат. Нормовані експоненційні втрати застосовуються для передбачення єдиного класу з K взаємно виключних класів. Сигмоїдні перехресно-ентропійні втрати застосовуються для передбачення K незалежних значень імовірності в проміжку [0,1]. Евклідові втрати застосовуються для регресії до дійснозначних міток (-∞, ∞).

1.7 Архітектури згорткових нейронних мереж

1.7.1 LeNet-5

LeNet-5 – це 7-рівнева згорткова нейронна мережа, була розгорнута в багатьох банківських системах для розпізнавання рукописних номерів на чеках.

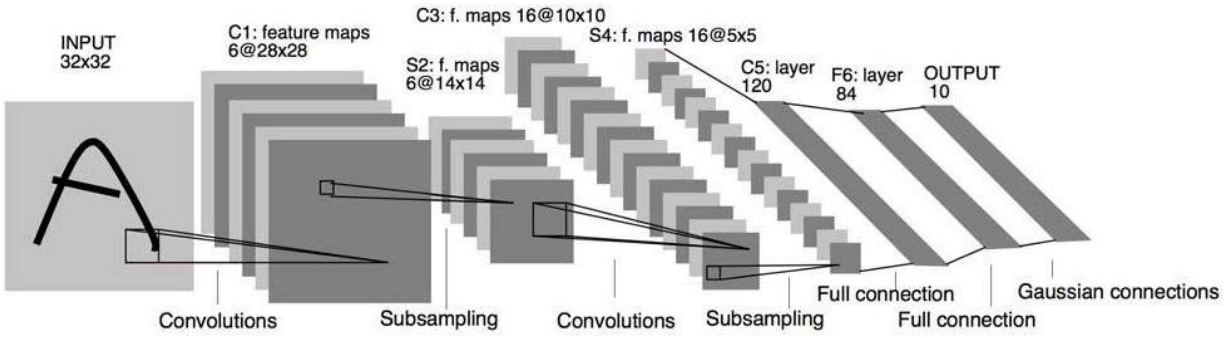

Рисунок 1.6 – Архітектура LeNet-5

За сучасними мірками, LeNet-5 - це дуже проста мережа. Вона містить лише 7 шарів, серед яких є 3 згорткові шари (C1, C3 і C5), 2 шари агрегування (S2 і S4) і 1 повноз'єднаний шар (F6), за яким слідує вихідний шар. Згорткові шари використовують 5x5 згортку з кроком 1. Шари агрегування використовують згортку 2x2. Активація сигмоїдною функцією використовується в усій мережі.

1.7.2 AlexNet

У 2012 році на конкурсі ILSVRC за класифікацією зображень вперше перемогла нейронна мережа - AlexNet, досягнувши top-5 помилки 15,31%. Для порівняння, метод, який не використовує свёрточние нейронні мережі, отримав помилку 26,1%. У AlexNet були зібрані новітні на той момент техніки для поліпшення роботи мережі.

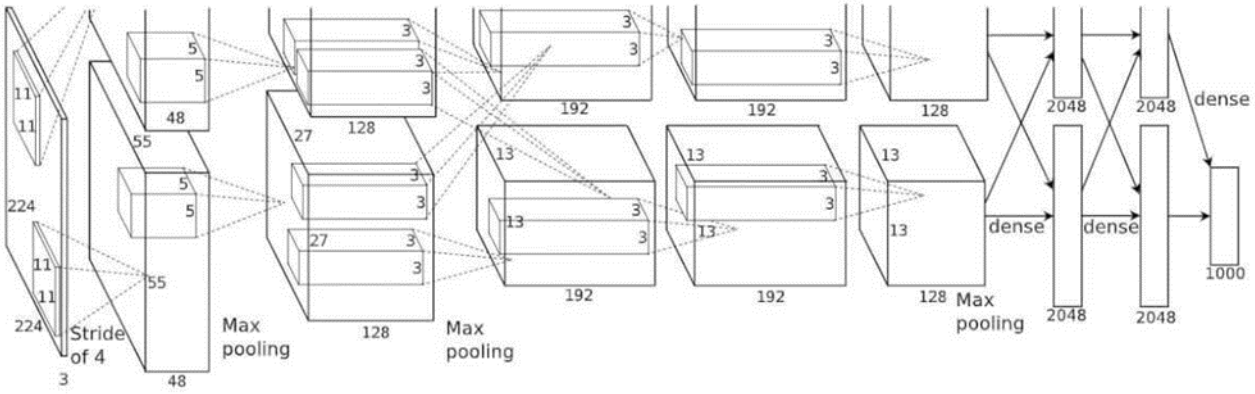

Рисунок 1.7 – Архітектура AlexNet

Навчання AlexNet через кількість параметрів мережі відбувалося на двох GPU, що дозволило скоротити час навчання в порівнянні з навчанням на CPU. Також виявилося, що використання функції активації ReLU (Rectified Linear Unit) замість більш традиційних функцій сигмоїд і гіперболічного тангенса дозволило знизити кількість епох навчання в шість разів.

Формула ReLU наступна: $y(x) = \max(0, x)$.

ReLU дозволяє побороти проблему загасання градієнтів, властиву іншим функціям активації.

Крім того, в AlexNet була застосована техніка відсіву (Dropout). Вона полягає у випадковому відключенні кожного нейрона на заданому шарі з ймовірністю p на кожній епосі. Після навчання мережі, на стадії розпізнавання, ваги шарів, до яких був застосований відсів, повинні бути помножені на 1/p. Відсів виступає в ролі регуляризатора, не дозволяючи мережі перенавчатися.

Для пояснення ефективності даної техніки існує кілька інтерпретацій. Перша полягає в тому, що відсів змушує нейрони не покладатися на сусідні нейрони, а навчатися розпізнавати більш стійкі ознаки. Друга, більш пізня, полягає в тому, що, навчання мережі з відсівом представляє собою апроксимацію навчання ансамблю мереж, кожна з яких представляє мережу без деяких нейронів. Таким чином, остаточне рішення приймає не одна мережа, а ансамбль, кожна мережа якого навчена по-різному, тим самим знижується ймовірність помилки.

Мережа була дуже схожа на LeNet, але була набагато глибшою і мала близько 60 мільйонів параметрів.

1.7.3 VGG Net

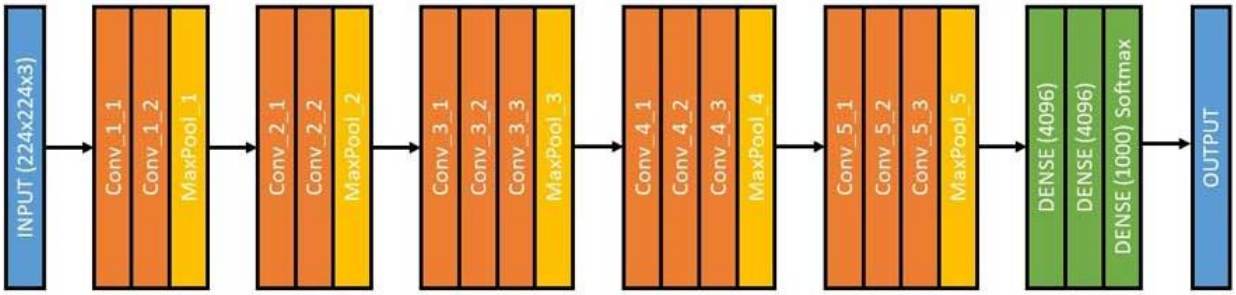

Рисунок 1.8 – Архітектура VGG Net

В даній мережі відмовилися від використання фільтрів розміром більше, ніж 3x3.

Автори показали, що шар з фільтром 7x7 еквівалентний трьом верствам з фільтрами 3x3, причому в останньому випадку використовується на 55% менше параметрів.

Аналогічно шар з фільтром 5x5 еквівалентний двом верствам з фільтром 3x3, які економлять 22% параметрів мережі.

На змаганні ILSVRC 2014 ансамбль з двох VGG Net отримав top-5 помилку 7,3%. Хоча дана модель і не перемогла в змаганні, через її простоти вона використовується в більш складних мережах, призначених для детектування предметів, семантичної сегментації або маскування об’єктів.

1.7.4 YOLOv3

YOLOv3 являється вдосконаленою версією архітектури YOLO. Вона складається з 106-ти згорткових шарів і краще виявляє невеликі об’єкти в порівнянні з її попередницею YOLOv2. Основна особливість YOLOv3 полягає в тому, що на виході є три шари кожен з яких розрахований на виявлення об'єктів різного розміру.

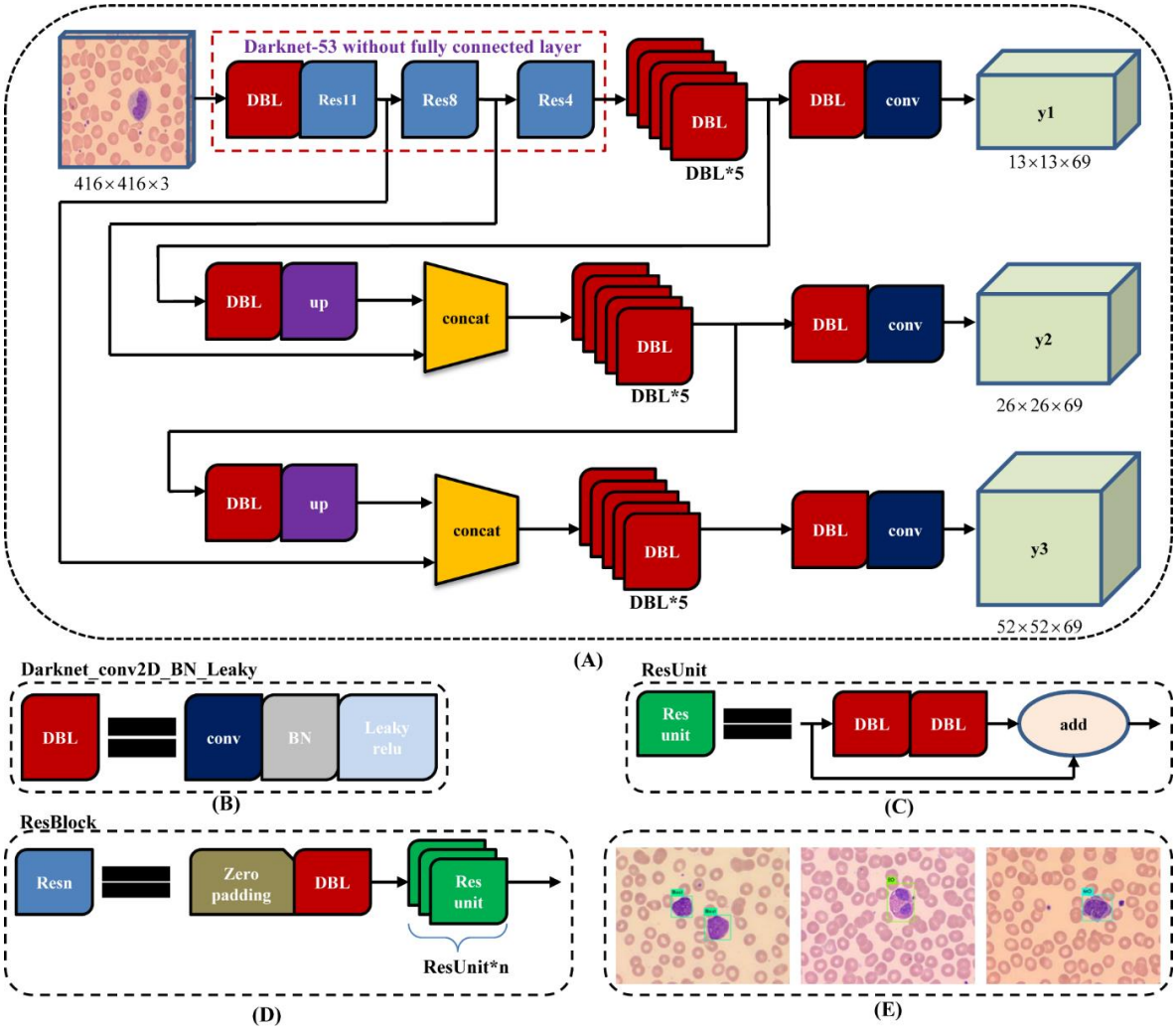

Рисунок 1.9 – Архитектура YOLOv3

1.8 Сучасні бібліотеки глибокого навчання

На сьогоднішній день існує більше десятка бібліотек для глибокого навчання. Ґрунтуючись на рекомендаційній інформації з широкого кола джерел, для подальшого розгляду було вибрано три бібліотеки: Tensorflow, Keras, PyTorch.

1.8.1 Tensorflow

Створена компанією Google і написана на Python і C ++, TensorFlow є однією з кращих відкритих бібліотек для глибокого навчання.

Google активно використовує цю бібліотеку для таких великомасштабних сервісів як Gmail і Google Translate. TensorFlow вже застосували до своїх сервісів і такі значні бренди як Uber, Airbnb, Dropbox і багато інших.

Переваги:

- для неї написано велику кількість посібників та документації;
- пропонує потужні засоби моніторингу процесу навчання моделей і візуалізації (Tensorboard);
- підтримується великою спільнотою розробників і технічними компаніями;
- забезпечує обслуговування моделей;
- підтримує розподілене навчання;
- забезпечує виведення на пристрої з низькою затримкою для мобільних пристроїв (TensorFlow Lite).

Недоліки:

- має більш високий вхідний поріг для початківців, ніж PyTorch або Keras;
- досить низькорівнева і вимагає багато шаблонного коду.

1.8.2 PyTorch

PyTorch – це портована на мову програмування Python бібліотека Torch, що була написана на Lua. Вона являється великим конкурентом TensorFlow. Бібліотека була розроблена компанією Facebook і використовувався Twitter, Salesforce, Оксфордським Університетом і багатьма іншими компаніями.

PyTorch використовується в основному, щоб навчати моделі швидко і ефективно, тому це хороший вибір для невеликих проектів.

Плюси:

- завдяки архітектурі фреймворку, процес створення моделі досить простий і прозорий;
- була розроблена для Python, і тому використовує його стандартні ідіоми;
- легше налагоджувати, тому що код виконується як звичайний Python код (немає етапу компіляції, як в TensorFlow);
- підтримує декларативний паралелізм даних;
- має багато попередньо навчених моделей і готових модульних частин, які легко комбінувати.

Мінуси:

- недостатня підтримка моделей;
- не підходить для великих проектів;
- бракує інтерфейсів для моніторингу та візуалізації;
- не підходить для кросплатформних проектів.

1.8.3 Keras

Keras – це мінімалістична бібліотека, написана на Python, яка може запускатися поверх TensoFlow, Theano або CNTK. Вона була розроблена інженером компанії Google, Франсуа Шолль, з метою прискорення

експериментів. Keras підтримує широкий спектр шарів нейронних мереж, таких як згорткові шари, рекурентні або щільні.

Плюси:

- прототипування дійсно швидке і просте;
- досить легкий для побудови моделей глибокого навчання для великої кількості шарів;
- має повністю конфігуровані модулі;
- простий і інтуїтивно-зрозумілий інтерфейс, відповідно, хороший для новачків;
- має вбудовану підтримку для навчання на декількох GPU;
- моделі можна навчати на кластерах GPU на платформі Google Cloud;
- підтримує можливість навчання на GPU від NVIDIA, TPU від Google та GPU з Open-CL, такі як AMD.

Мінуси:

- занадто високорівневий для великих та складних проектів і не завжди легко кастомізується;
- обмежений бекендами Tensorflow, CNTK і Theano.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Опис вхідних даних

Вхідними даними до задачі класифікації емоцій є датасет FER2013. Дані складаються із 35887 чорно-білих зображень обличчя в масштабі 48x48 пікселів. Обличчя були автоматично зареєстровані так, що обличчя більш-менш відцентроване і займає приблизно однакову кількість місця у кожному зображенні.

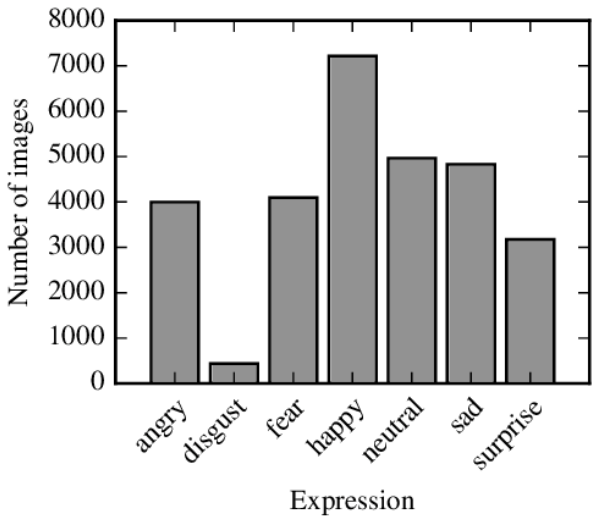

Рисунок 2.1 – Гістограма частот екземплярів FER2013

Кожному з зображень було присвоєно мітку однієї з семи емоцій: 0 – гнів, 1 – огида, 2 – страх, 3 – щастя, 4 – сум, 5 – здивування, 6 – нейтральний.

Після аналізу вибірки і побудови гістограми частот екземплярів (рис. 2.1), можна побачити, що вона є надто нерівномірною. Деякі екземпляри представлені недостатньою кількістю екземплярів. Щоб не допустити перенавчання моделі на екземплярах, що зустрічаються частіше, можливо, знадобиться вирівняти датасет шляхом генерації додаткових екземплярів.

2.2 Опис вихідних даних

Вихідними даними задачі є клас емоції, зображення якого було «скормлено» нейронній мережі, і яке вона до того не бачила. Таким чином буде побудована модель, що здатна класифікувати емоції 7 класів, що представлені в навчальній вибірці.

2.3 Розмірність задачі

Таким чином, маємо датасет емоцій, розміри якого в першу чергу визначають розмірність задачі.

Ключова інформація про дані наведена у таблиці 1.

Таблиця 2.1 – Основні статистичні дані

Об’єм навчальної вибірки	28709
Об’єм валідаційної вибірки	3589
Об’єм тестової вибірки	3589
Розмірність екземпляра (зображення)	48x48x1
Кількість класів	7

2.4 Архітектура нейронної мережі

Зважаючи на наведений вище огляд типових архітектур нейронних мереж і на клас задачі, що вирішується в даній роботі (задача класифікації зображень), архітектура нейронної мережі буде спроектована наступним чином: поєднаймо згорткову нейронну мережу для вилучення ознак і повнозв’язний перцептрон для класифікації екземплярів.

Спроектована нейронна мережа приймає на вхід зображення, представлене масивом цілих чисел розмірністю 48x48x1. Такий масив містить 35877 елементів. Згорткова нейронна мережа дозволяє вилучити з зображення його ключовими ознаками, по яких можна класифікувати або розпізнати ті чи інші об’єкти. Емпірично встановлено, що для даного формату вхідних даних згорткові фільтри розміром 3x3 дозволяють виділити основні характеристики зображення, не пропустивши важливих ознак. Застосуємо декілька таких фільтрів до вхідного зображення.

Для того щоб мінімізувати кількість параметрів нейромережі і виділити тільки найважливіші ознаки, застосовується пулінг. Візьмемо розмірність сітки пулінгу 3x3.

2.4 Функція активації

У якості активаційних функцій доцільно обрати будь-які неспадні дифиренційовні функції, що діють на множині дійсних чисел. Можливі варіанти можуть бути такі: лінійна функція з насиченням, ReLU, сигмоїд, тангенс гіперболічний, порогова функція. Деякі з них зображені на рис. 2.1.

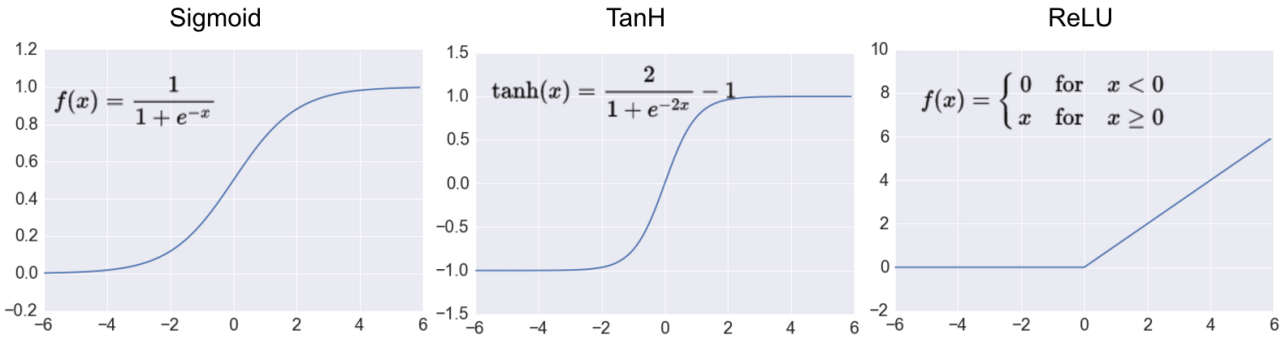

Рисунок 2.2 – Графіки основних функцій активації

Використаємо ReLU: $h(a) = \max(0, a)$, де $a = WX + b$. Застосування ReLU забезпечує дві ключові переваги, що дозволяють пришвидшити навчання

нейромережі: розрідженість та менша ймовірність розмиття градієнту в порівнянні з використанням інших активаційних функцій.

Розрідженість виникає, коли a < 0. Чим більша кількість нейронів з ReLU активацією в шарі, тим більша розрідженість результуючого представлення. Сигмоїд же, наприклад, завжди повертає деяке ненульове значення, що призводить до «згущення» представлення. З точки зору швидкості навчання розріджене представлення завжди краще за «згущене».

2.5 Нормалізація вихідних даних

Побудована за такою схемою мережа має на своїх виходах деякі числа. Кожен вихід характеризує вхідний екземпляр відповідно конкретному класу. Для підвищення інтерпретованості моделі є сенс виразити отримані величини у ймовірнісній формі, так, щоб модель давала відповідь на питання: до якого классу найімовірніше належить вхідний екземпляр. Для цього до виходів повнозв'язного перцептрона підключимо Softmax-функцію (функція експоненційної нормалізації). Така функція дозволяє перетворити деякий k-мірний вектор z в k-мірний вектор s(z) дійсних чисел на проміжку від 0 до 1, такий що сума його елементів завжди дорівнює 1. Формула перетворення j-го елементу наведена нижче:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

2.6 Діаграма потоків даних

Для досягнення кінцевої цілі, розпізнавання емоцій за зображенням, треба зробити наступні кроки: 1) зчитати зображення з веб-камери; 2) обробити вхідне

зображення; 3) виявити обличчя на зображенні; 4) класифікувати емоцію; 5) вивести вихідне зображення на екран монітору.

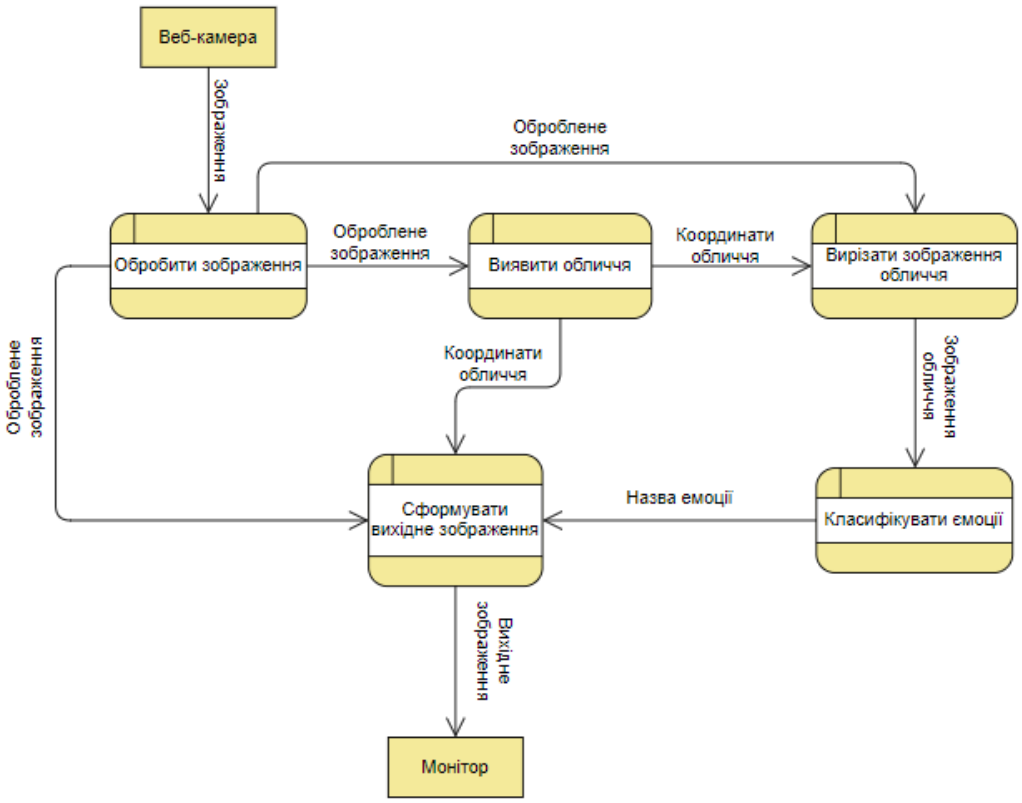

Рисунок 2.3 – Діаграма потоків даних

Виходячи з інформації описаною мною вище, розроблювана система, згідно з нотацією DFD буде мати наступні елементи:

– веб-камера, яка являється джерелом даних;
– процес обробки вхідного зображення (бінарізація зображення, виділення границь тощо);
– процес виявлення обличчя;
– процес вилучення зображення обличчя;
– процес класифікації емоцій;
– процес формування вихідного зображення;
– екран монітору, що являється зовнішньою сутністю, одержувачем вихідної інформації.

3 РОЗРОБКА ТА ТЕСТУВАННЯ

3.1 Використане обладнання та програмне забезпечення

Розробка всіх основних компонентів системи, реалізація класифікаторів, навчання, тестування та валідація класифікаторів проводилася на комп'ютері під керуванням операційної системи Ubuntu.

Для навчання, тестування і перевірки моделі класифікатору використовувався центральний процесор Intel Core i3-4000m.

Для розробки даної системи, була обрана мова програмування Python. Оскільки існує велика кількість бібліотек з відкритим кодом для глибокого навчання, що написані на мові програмування Python.

Для створення та навчання моделі використовувалася відкрита нейромережева бібліотека Keras. Вона представляє високорівневий, інтуїтивний набір абстракцій, який робить простим формування нейронних мереж та їх навчання.

Для того щоб створена система могла в реальному часі класифікувати емоції на відепотоці, була використана бібліотека OpenCV.

Для задачі виявлення обличчя буде використовуватися бібліотеку OpenCV. Це бібліотека з відкритим кодом, в неї входять велика кількість функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на відепотоці.

3.2 Параметри нейронної мережі

Початкова архітектура складалася з дев'яти згорткових шарів та повнозв'язного шару у кінці. Але пізніше було прийнято рішення видалити

останній повнозв’язний та змінити повнозв’язні шари на глибоко відокремлювані згортки, подібні на ті що використовуються в нейронній мережі Xception. Що дозволило суттєво зменшити кількість параметрів та пришвидшити процес класифікації на апаратно обмежених системах.

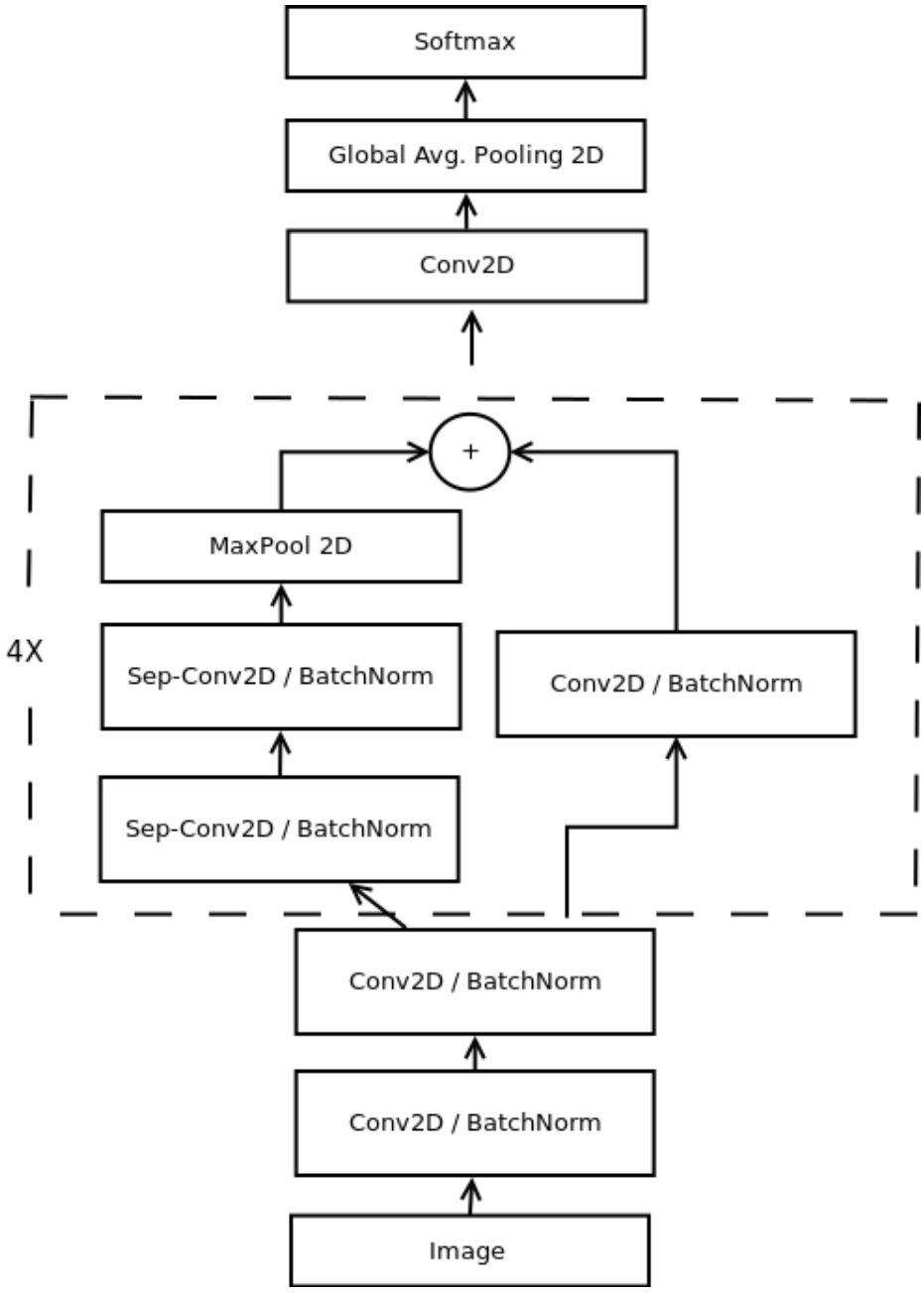

Рисунок 3.1 – Архітектура кінцевої нейронної мережі

Кінцева архітектура нейронної мережі – це повністю згортока нейронна мережа, що складається з чотирьох залишкових глибоко-відокремлюваних згорток. Після кожного згорткового шару застосовується операція нормалізації

BatchNormalization та функція активації ReLU. Останній шар застосовує global average pooling та функцію активації softmax для створення прогнозу.

Ця архітектура має приблизно 60 000 параметрів, що у десятки разів менше ніж у початковій архітектурі.

3.3 Навчання нейронної мережі

Навчання нейронної мережи проводилось на центральному процесорі Intel Core i3-4000m з оптимізатором ADAM.

Валідаційну вибірка для контролю процесу навчання формувалася виходячи з відношення 1/8 від навчальної вибірки. Точність класифікатора на валідаційній вибірці досягла 66% на 102 епосі, після чого почала знижуватися. Навчання було зупинено на 105 епосі.

3.4 Результати випробування системи

Контрольна вибірка формувалися виходячи з відношення 1/8 від навчальної вибірки.

Результати класифікації на контрольній вибірці виявилися хорошими і цілком передбачуваними. Точність класифікатора склала 66% на контрольній вибірці. Але показник точності для різних міток суттєво відрізняється. Наприклад для класифікації страху точність складає всього 41%. Цю різницю можна побачити на матриці неточностей, що зображена на рисунку 3.2.

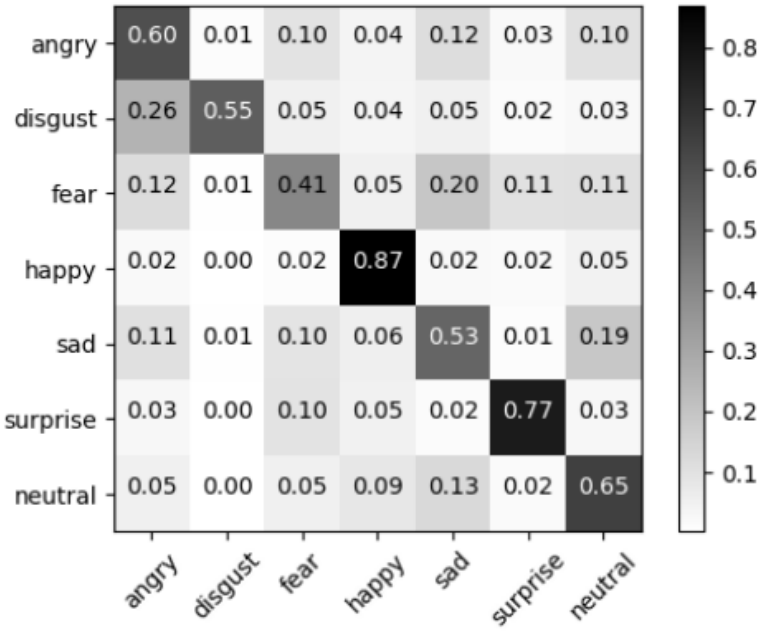

Рисунок 3.2 – Матриця неточностей

Різниця в показниках пов’язана з тим, що навчальна вибірка досить неоднорідна, для деяких із міток було досить мало навчальних даних.

3.5 Тестування системи в реальному часі

Оскільки кінцевою метою було створення саме системи розпізнавання емоцій в реальному часі, то за допомогою бібліотеки OpenCV, можна легко зчитати зображення з веб-камери та виявити обличчя на ньому. Дана бібліотека вже містить усі потрібні для нашої цілі функції для обробки зображення і багато заздалегідь підготовлених детекторів для обличчя, очей, посмішок тощо.

Все що потрібно зробити – це завантажити навчений каскад Хаара для виявлення обличь (haarcascade_frontalface_default.xml).

Обов’язково треба конвертувати зображення в чорно-біле, оскільки детектор обличь і розроблений класифікатор працюють лише з чорно-білими зображеннями.

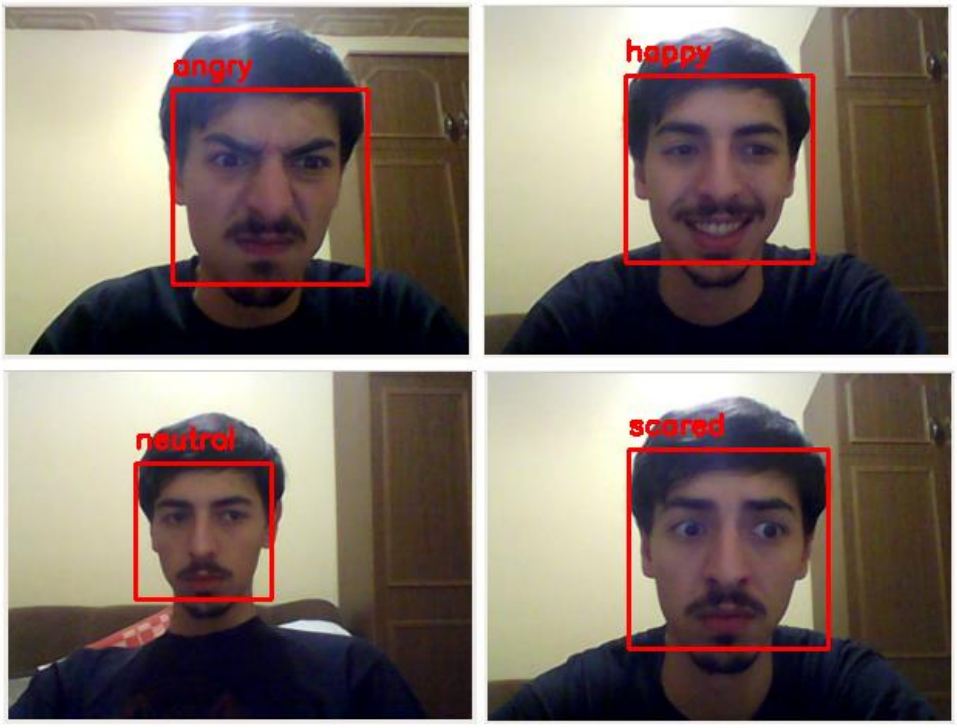

Рисунок 3.3 – Результати роботи системи в реальному часі

На рисунку 3.3 можна побачити результат роботи класифікатора в реальному часі.

ВИСНОВКИ

При виконанні кваліфікаційної роботи було розглянуто засоби виявлення та розпізнавання образів на зображеннях системами комп’ютерного зору, основні методи та задачі комп’ютерного зору, основні бібліотеки для створення штучних нейронних мереж та глибокого навчання.

Проаналізувавши інструментарій та мови програмування, для розробки програмного продукту було обрано мову програмування Python. Як бібліотеку для глибокого навчання було обрано Keras. Для обробки зображення та виявлення образів на зображенні було використано бібліотеку OpenCV.

В процесі розробки дипломного проекту було вивчено та порівняно декілька архітектур згорткових нейронних мереж. Для реалізації обрано метод який найкраще відповідає вимогам та поставленим задачам.

Результатом кваліфікаційної роботи став програмний продукт, який дозволяє обробляти кадри з відеокамери, виявляти обличчя та класифікувати емоції.

ПЕРЕЛІК ПОСИЛАНЬ

1. Форсайт Д.А., Понс Ж. Комп’ютерний зір. Сучасний підхід. Москва: Вильямс, 2004. 928 с

2. Уоссермен Ф. Нейрокомпьютерна техніка: Теорія і практика. Москва : Мир, 1992. 184 с.

3. Хайкін С. Нейронні мережі: повний курс, 2-е видання. Москва : Вильямс, 2008. 1103 с.

4. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. URL: https://arxiv.org/pdf/1506.02640.pdf (дата звернення: 01.11.2019).

5. LeCun Y., Bottou L., Bengio Y., Haffner P. GradientBased Learning Applied to Document Recognition. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf (дата звернення: 07.11.2019).

6. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. URL: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (дата звернення: 07.11.2019).

7. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. URL: https://arxiv.org/pdf/1409.1556.pdf (дата звернення: 08.11.2019).

8. Nielsen M. Neural Networks And Deep Learning. URL: http://neuralnetworksanddeeplearning.com/ (дата звернення: 10.12.2019).

ДОДАТОК А

Код програми

```
def XNET(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
                                    use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
                                    use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1
    residual = Conv2D(16, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
```

```
x = SeparableConv2D(16, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 2
residual = Conv2D(32, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 3
residual = Conv2D(64, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
```

```
x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 4
residual = Conv2D(128, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(128, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(128, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
```

```
    x = layers.add([x, residual])

    x = Conv2D(num_classes, (3, 3),
           padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    output = Activation('softmax',name='predictions')(x)

    model = Model(img_input, output)
    return model


from keras.preprocessing.image import img_to_array
import imutils
import cv2
from keras.models import load_model
import numpy as np

# parameters for loading data and images
detection_model_path = 'haarcascade_files/haarcascade_frontalface_default.xml'
emotion_model_path = 'models/_mini_XCEPTION.102-0.66.hdf5'

# hyper-parameters for bounding boxes shape
# loading models
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
EMOTIONS = ["angry" ,"disgust","scared", "happy", "sad", "surprised",
 "neutral"]

# starting video streaming
cv2.namedWindow('your_face')
```

```
camera = cv2.VideoCapture(0)
while True:
    frame = camera.read()[1]
    #reading the frame
    frame = imutils.resize(frame,width=300)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces =
face_detection.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(30,
30),flags=cv2.CASCADE_SCALE_IMAGE)

    canvas = np.zeros((250, 300, 3), dtype="uint8")
    frameClone = frame.copy()
    if len(faces) > 0:
        faces = sorted(faces, reverse=True,
        key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
        (fX, fY, fW, fH) = faces
        roi = gray[fY:fY + fH, fX:fX + fW]
        roi = cv2.resize(roi, (64, 64))
        roi = roi.astype("float") / 255.0
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis=0)
        preds = emotion_classifier.predict(roi)[0]
        emotion_probability = np.max(preds)
        label = EMOTIONS[preds.argmax()]
    else: continue


    for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
            # construct the label text
            text = "{}: {:.2f}%".format(emotion, prob * 100)
```

```
            w = int(prob * 300)
            cv2.rectangle(canvas, (7, (i * 35) + 5),
            (w, (i * 35) + 35), (0, 0, 255), -1)
            cv2.putText(canvas, text, (10, (i * 35) + 23),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45,
            (255, 255, 255), 2)
            cv2.putText(frameClone, label, (fX, fY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
            cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH),
                    (0, 0, 255), 2)

    cv2.imshow('your_face', frameClone)
    cv2.imshow("Probabilities", canvas)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

camera.release()
cv2.destroyAllWindows()



from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from load_and_process import load_fer2013
from load_and_process import preprocess_input
from models.cnn import mini_XCEPTION
from sklearn.model_selection import train_test_split

# parameters
batch_size = 32
```

```
num_epochs = 10000
input_shape = (48, 48, 1)
validation_split = .2
verbose = 1
num_classes = 7
patience = 50
base_path = 'models/'

# data generator
data_generator = ImageDataGenerator(
                    featurewise_center=False,
                    featurewise_std_normalization=False,
                    rotation_range=10,
                    width_shift_range=0.1,
                    height_shift_range=0.1,
                    zoom_range=.1,
                    horizontal_flip=True)

# model parameters/compilation
model = mini_XCEPTION(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
            metrics=['accuracy'])
model.summary()

# callbacks
log_file_path = base_path + '_emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                              patience=int(patience/4), verbose=1)
```

```
trained_models_path = base_path + '_mini_XCEPTION'
model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
                                   save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

# loading dataset
faces, emotions = load_fer2013()
faces = preprocess_input(faces)
num_samples, num_classes = emotions.shape
xtrain, xtest,ytrain,ytest = train_test_split(faces, emotions,test_size=0.2,shuffle=True)
model.fit_generator(data_generator.flow(xtrain, ytrain,
                                        batch_size),
                    steps_per_epoch=len(xtrain) / batch_size,
                    epochs=num_epochs, verbose=1, callbacks=callbacks,
                    validation_data=(xtest,ytest))
```