МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ ПСИХОЕМОЦІЙНОГО СТАНУ ЛЮДИНИ В СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ»**

Виконав: студент 2 курсу, групи 8.1224
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
І.В. Дубровний
(ініціали та прізвище)
Керівник ректор ЗНУ, д.т.н., професор Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент професор кафедри програмної інженерії, к.ф.-м.н., доцент Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " 12 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ СТУДЕНТОВІ

Дубровному Іллі Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Дослідження технологій розпізнавання психоемоційного стану людини в системах комп'ютерного зору

керівник роботи Шило Галина Миколаївна, д.т.н., професор
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » 02 2025 року № 326-с

2. Строк подання студентом роботи 03.12.2025

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. Проектування системи та методологія порівняльного аналізу.
3. Реалізація та тестування системи розпізнавання емоцій.

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 03.03.2025

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.	05.03.2025	
2.	Збір вихідних даних.	10.04.2025	
3.	Обробка методичних та теоретичних	12.05.2025	
	джерел.		
4.	Розробка першого та другого розділу.	15.07.2025	
5.	Розробка третього розділу.	10.10.2025	
6.	Оформлення та нормоконтроль	01.12.2025	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	17.12.2025	

Студент _____ І.В. Дубровний
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.М. Шило
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Дослідження технологій розпізнавання психоемоційного стану людини в системах комп'ютерного зору»: 77 с., 15 рис., 10 формул, 23 джерела, 4 додатка.

БЕНЧМАРКІНГ, ДОНАВЧАННЯ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КВАНТУВАННЯ, КОМП'ЮТЕРНИЙ ЗІР, ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ, ПЕРЕНЕСЕННЯ НАВЧАННЯ, РОЗПІЗНАВАННЯ ЕМОЦІЙ, СИСТЕМИ РЕАЛЬНОГО ЧАСУ.

Об'єкт дослідження – процес автоматизованого розпізнавання психоемоційного стану людини за зображенням обличчя з використанням методів комп'ютерного зору та машинного навчання.

Мета роботи: підвищення ефективності систем розпізнавання емоцій шляхом розробки, навчання та програмної оптимізації моделі на базі архітектури MobileNetV2, що дозволить досягти балансу між високою точністю класифікації та швидкодією, необхідною для роботи в режимі реального часу.

Методи дослідження – аналітичний, експериментальний.

У ході дослідження проаналізовано методи комп'ютерного зору та реалізовано нейронну мережу на базі архітектури MobileNetV2 із застосуванням Transfer Learning на датасеті FER-2013. Виконано оптимізацію моделі шляхом динамічного квантування у формат TensorFlow Lite. Експериментально підтверджено досягнення швидкодії понад 64 FPS на CPU, що перевершує показники бібліотеки DeepFace при співставній точності. Отримані результати рекомендовано для створення систем реального часу на пристроях з обмеженими ресурсами.

SUMMARY

Master’s Qualifying Theses «Research into technologies for recognizing a person’s psycho-emotional state in computer vision systems»: 77 pages, 15 figures, 10 formulas, 23 references, 4 supplements.

BENCHMARKING, FINE-TUNING, CONVOLUTIONAL NEURAL NETWORKS, QUANTIZATION, COMPUTER VISION, PERFORMANCE OPTIMIZATION, TRANSFER LEARNING, EMOTION RECOGNITION, REAL-TIME SYSTEMS.

Object of the study – the process of automated recognition of a person’s psycho-emotional state based on facial images using computer vision and machine learning methods.

Aim of the study: to improve the efficiency of emotion recognition systems by developing, training and software optimizing a model based on the MobileNetV2 architecture, which will achieve a balance between high classification accuracy and the speed required for real-time operation.

Methods of research – analytical and experimental.

The study analyzed computer vision methods and implemented a neural network based on the MobileNetV2 architecture using Transfer Learning on the FER-2013 dataset. The model was optimized by dynamic quantization into the TensorFlow Lite format. Experimental results confirmed that the CPU achieved a performance of over 64 FPS, which exceeds the performance of the DeepFace library with comparable accuracy. The results obtained are recommended for creating real-time systems on devices with limited resources.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 9
1 Аналіз предметної області та існуючих рішень 11
1.1 Психофізіологічні основи емоцій 11
1.1.1 Теорія базових емоцій (Категоріальний підхід) 11
1.1.2 Анатомічна декомпозиція: система кодування лицьових рухів (FACS) 13
1.1.3 Дименсійна модель (Багатовимірний підхід) 16
1.1.4 Статичні та динамічні ознаки обличчя 17
1.1.5 Постановочні та спонтанні емоції 18
1.2 Огляд методів комп'ютерного зору для розпізнавання емоцій 19
1.2.1 Традиційні методи 19
1.2.2 Методи глибокого навчання (Deep Learning) 21
1.2.3 Огляд ключових архітектур для порівняльного аналізу 22
1.2.4 Перенос навчання 23
1.3 Огляд існуючих наборів даних (Datasets) 23
1.3.1 Лабораторні набори даних 24
1.3.2 Природні набори даних 24
1.4 Аналіз готових рішень та бібліотек 25
1.4.1 Open-Source бібліотеки (Python) 25
1.4.2 Хмарні сервіси (Cloud API) 26
2 Проектування системи та методологія порівняльного аналізу 27
2.1 Обґрунтування вибору технологій для дослідження 27
2.1.1 Каскадний підхід – бібліотека FER (MTCNN) 27
2.1.2 Глибокі мережі – DeepFace (VGG-Face) 28

2.1.3 Ефективна архітектура – MobileNet 28
2.2 Характеристика програмно-апаратного комплексу дослідження 29
2.2.1 Апаратне забезпечення 29
2.2.2 Системне та інструментальне програмне забезпечення 30
2.2.3 Бібліотеки машинного навчання та обробки даних 30
2.3 Методика підготовки даних та алгоритм тестування 31
2.3.1 Характеристика набору даних (FER-2013) 32
2.3.2 Алгоритм попередньої обробки (Preprocessing Pipeline) 32
2.3.3 Протокол тестування 34
2.4 Система метрик для оцінки ефективності алгоритмів 35
2.4.1 Метрики якості розпізнавання 35
2.4.2 Метрики обчислювальної ефективності 36
3 Реалізація та тестування системи розпізнавання емоцій 38
3.1 Підготовка та налаштування моделей 38
3.1.1 Готові моделі (Out-of-the-box) 38
3.1.2 Підготовка моделі MobileNetV2 38
3.1.3 Розробка програмного модуля для інференсу навченої моделі 41
3.2 Розробка програмного засобу для порівняльного аналізу 44
3.2.1 Структура та алгоритм роботи програми 44
3.2.2 Результати порівняльного аналізу 46
3.3 Оптимізація моделі з використанням TensorFlow Lite 47
3.3.1 Розробка конвертера та калібрування 48
3.3.2 Реалізація універсального модуля інференсу 48
3.3.3 Результати оптимізації 49
Висновки 51
Перелік посилань 53
Додаток А Код підпрограми для навчання моделі 56
Додаток Б Код підпрограми для тестування навченої моделі на користувацьких зображеннях 63
Додаток В Код основного програмного модуля для бенчмаркінгу 66
Додаток Г Код підпрограми для конвертації моделі у формат TFLite 77

ВСТУП

В наш час системи комп’ютерного зору та штучного інтелекту — це галузі, що динамічно розвиваються та знаходять застосування у все більшій кількості сфер людського життя. Нові архітектури, моделі та підходи до аналізу даних з’являються чи не щодня, і важливо бути в курсі цих змін, щоб створювати ефективні та інноваційні інтелектуальні системи. Автоматичне розпізнавання психоемоційного стану людини — це лише один з багатьох напрямків, що здатний кардинально поглибити рівень взаємодії людини з комп’ютером та навіть надати доступ для нових видів цієї самої взаємодії у різноманітних галузях.

Використання технологій розпізнавання емоцій має декілька ключових переваг. По-перше, це дозволяє створювати по-справжньому адаптивні та емпатичні системи, які можуть реагувати на настрій користувача, наприклад, в освітніх платформах чи медичних додатках. По-друге, аналіз емоцій може значно підвищити якість продуктів та послуг (через аналіз реакції клієнтів) або рівень безпеки (наприклад, моніторинг втоми водія).

Розробка та дослідження систем розпізнавання емоцій — це складний процес, який вимагає глибоких знань у сфері комп’ютерного зору, машинного навчання та розуміння психофізіологічних маркерів емоцій. Тому для досягнення мети кваліфікаційної роботи важливо детально розглянути та проаналізувати такі аспекти:

- огляд теоретичних основ та моделей психоемоційних станів, а також підходів до їх класифікації в системах комп’ютерного зору (аналіз міміки, мікроекспресій, рухів голови). У цьому розділі будуть розглянуті базові теорії (наприклад, моделі Екмана, Рассела) та те, як вони трансформуються у ознаки, зрозумілі для машин;

- аналіз сучасних технологій та архітектур глибокого навчання (наприклад, DeepFace, FER), які використовуються для розпізнавання емоцій за зображенням обличчя;
- розробка методики та проведення порівняльного аналізу декількох обраних готових технологій (навчених моделей або API). У цьому розділі буде описано процес тестування обраних моделей на контрольному наборі даних за ключовими метриками (точність, швидкодія, F1-score) для виявлення їхніх переваг та недоліків.

Метою цієї роботи є дослідження та порівняльний аналіз ефективності сучасних технологій комп’ютерного зору для розпізнавання психоемоційного стану людини. Для реалізації цієї мети будуть проаналізовані різні підходи, обрані ключові моделі для експериментального тестування та сформульовані висновки щодо їх практичної придатності для вирішення прикладних задач.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Психофізіологічні основи емоцій

Психоемоційний стан людини – це складний процес, що супроводжується деякими спеціальними реакціями організму, а ці реакції, в свою чергу, можна зареєструвати та проаналізувати. Для систем комп'ютерного зору найбільш важливим аспектом є зовнішній прояв емоцій, або експресія: міміка обличчя людини є одним з найінформативніших каналів невербальної комунікації.

Розуміння психологічних моделей емоцій є критично важливим етапом для створення інтелектуальних систем, які розраховані на роботу з комп'ютерним зором, бо архітектура за метод розмітки даних залежать від обраної теоретичної бази. В наш час в цьому напрямку виділяють два різних підходи: категоріальний (теорія базових емоцій) та вимірний підхід (дименсійна модель).

1.1.1 Теорія базових емоцій (Категоріальний підхід)

Цей підхід базується на дослідженнях Чарльза Дарвіна та Пола Екмана. Перший у своїй роботі «Вираження емоцій у людини і тварин» (1872) висунув гіпотезу про те, що мімічні рухи у людей – це просто рудименти нормальних біологічних реакцій. Екман в 1970-х роках експериментально підтвердив гіпотезу про те, що емоції є універсальними: завдяки проведенню досліджень в різноманітних культурних групах (навіть в ізольованих племенах), він довів, що існують певні вирази обличчя, які інтерпретуються однаково не залежно від раси, культури чи мови.

Пол Екман виділив шість основних емоцій (див. рис. 1.1), які мають свої візуальні шаблони:

– радість, яка відчувається при досягненні цілей чи задоволенні якихось потреб. Характеризується скороченням великого виличного м’яза (підтягування куточків рота вгору) та колового м’яза ока (поява невеликих зморшок біля очей);
– сум, який пов’язаний з втратою чогось, пригніченням. Зазвичай характеризується піднятими внутрішніми куточками брів, опущеними вниз куточками губ, розфокусованим поглядом;
– страх, як реакція на небезпеку, що супроводжується загостренням уваги та прискореним серцебиттям. Характеризується тим, що брови підняті та зведені разом, верхні повіки підняті так, що оголяється склера, а губи напружені та розтягнуті;
– гнів, почуття роздратованості чи злості. Характеризується так: брови опущені та зведені, губи щільно стиснуті або оголюють зуби, погляд пильний, м’язи загалом напружені;
– здивування: швидка реакція на несподіванку, що може бути як позитивною, так і негативною. Характеризується високо піднятими бровами, широко відкритими очами, опущеною та розслабленою щелепою;

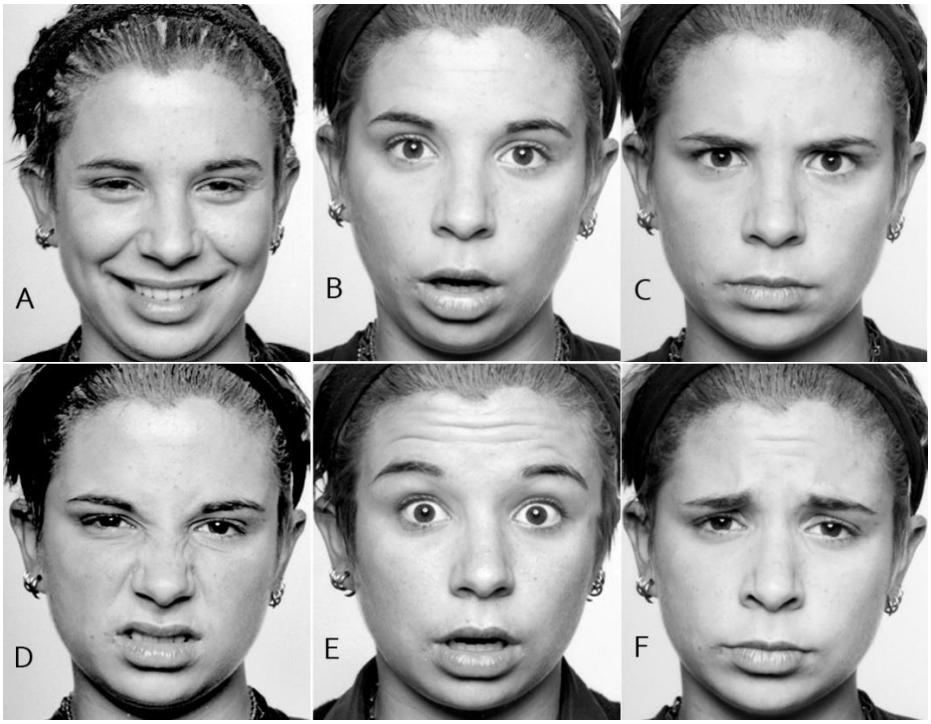


Рисунок 1.1 – Базові емоції Пола Екмана: A – радість, B – здивування, C – гнів, D – відраза, E – страх, F – сум

– відраза, захисна реакція на те, що сприймається як щось небажане, шкідливе. Характеризується зморщеним носом та піднятою верхньою губою.

Також до цього переліку часто додають зневагу, яка характеризується асиметричною напругою куточка рота, і так званий «нейтральний стан».

Якщо розглядати все це з боку реалізації системи комп'ютерного зору, цей підхід зводить завдання до задачі багатокласової класифікації, для якої результатом буде ймовірність приналежності зображення людини до одного з дискретних класів, або емоцій [1,2].

1.1.2 Анатомічна декомпозиція: система кодування лицьових рухів (FACS)

Хоча теорія базових емоцій і була доволі популярною, подальші дослідження показали, що простої класифікації на 6-7 категорій недостатньо для повноцінного аналізу людської поведінки. По-перше, дискретні категорії занадто суб'єктивні, один і той же візуальний сигнал (наприклад насунуті брови) може бути інтерпретований і як гнів, і як зосередженість, або навіть як відчуття фізичного болю. По-друге, дуже часто людина відчуває комбіновані емоції (наприклад сумна посмішка, або здивування разом зі страхом), і побачити варіації у «чистому» вигляді та класифікувати їх в якусь одну певну категорію важко. І по-третє, така класифікація не враховує інтенсивність емоцій: наприклад легка роздратованість та неконтрольована лють підпадають під класифікацію «Гнів», хоча візуально та фізіологічно дуже сильно відрізняються.

Тому для об'єктивного вимірювання міміки людини, виключаючи суб'єктивні моменти (наприклад як інтерпретація спостерігача), Пол Екман та Уоллес Фрізен розробили систему кодування лицьових рухів, або FACS (Facial

Action Coding System). Це анатомічно орієнтована система, яка описує візуально помітні рухи обличчя.

Основна ідея FACS полягає у переході від інтерпретації почуттів людини (як у системі базових емоцій Пола Екмана) до об'єктивного опису того, що можна побачити. Система базується на анатомії лицьових м'язів, і для кожного візуально помітного руху були присвоєні унікальні коди – Одиниці руху (Action Units, AU).

Перевагами такого підходу є:

– об'єктивність: кодування не залежить від контексту або інтерпретації можливого спостерігача. Якщо куточки губ підняті – це завжди AU 12, незалежно від того, чи щира це посмішка, або злісний вищир;
– детальність: ця система дозволяє описати більше 10000 різноманітних комбінацій виразів обличчя, в той час як теорія базових емоцій підтримує лише декілька категорій.

Кожна класифікація одиниці руху пов'язана зі скороченням певного м'язу або групи м'язів. Це є критично важливим для систем комп'ютерного зору, бо це дозволяє алгоритмам відстежувати зміщення контрольних точок, які відповідають цим м'язам, на обличчі. Як приклад, можна навести такі ключові AU:

– AU 1 (Inner Brow Raiser): підняття внутрішньої частини брови (лобовий м'яз, m. frontalis pars medialis). Є ключовим маркером смутку і стресу, який вкрай складно імітувати свідомо;
– AU 9 (Nose Wrinkler): зморщування носа (m. levator labii superioris alaeque nasi). Є основною ознакою відрази;
– AU 12 (Lip Corner Puller): відтягування куточків рота вгору (m. zygomaticus major). Одна зі складових посмішки.

Саме FACS дозволяє науковими методами обґрунтувати різницю між складними але схожими виразами обличчя, що є найважливішим завданням системи комп'ютерного зору. Як класичний приклад можна розглянути

різницю між щирою посмішкою (посмішка Дюшена) та соціальною (не щирою) посмішкою (див. рис. 1.2):

– щира посмішка: обов’язкова комбінація AU 12 + AU 6. AU 6 (Cheek Raiser) відповідає за скорочення кругового м’яза ока (m. orbicularis oculi), що викликає підняття щік та появу зморшок навколо очей. Цей м’яз слабо піддається свідомому контролю, тому здебільшого сигналізує про щиру емоцію;
– соціальна посмішка: активується тільки AU 12, людина просто розтягує губи, що можна легко зробити свідомо.

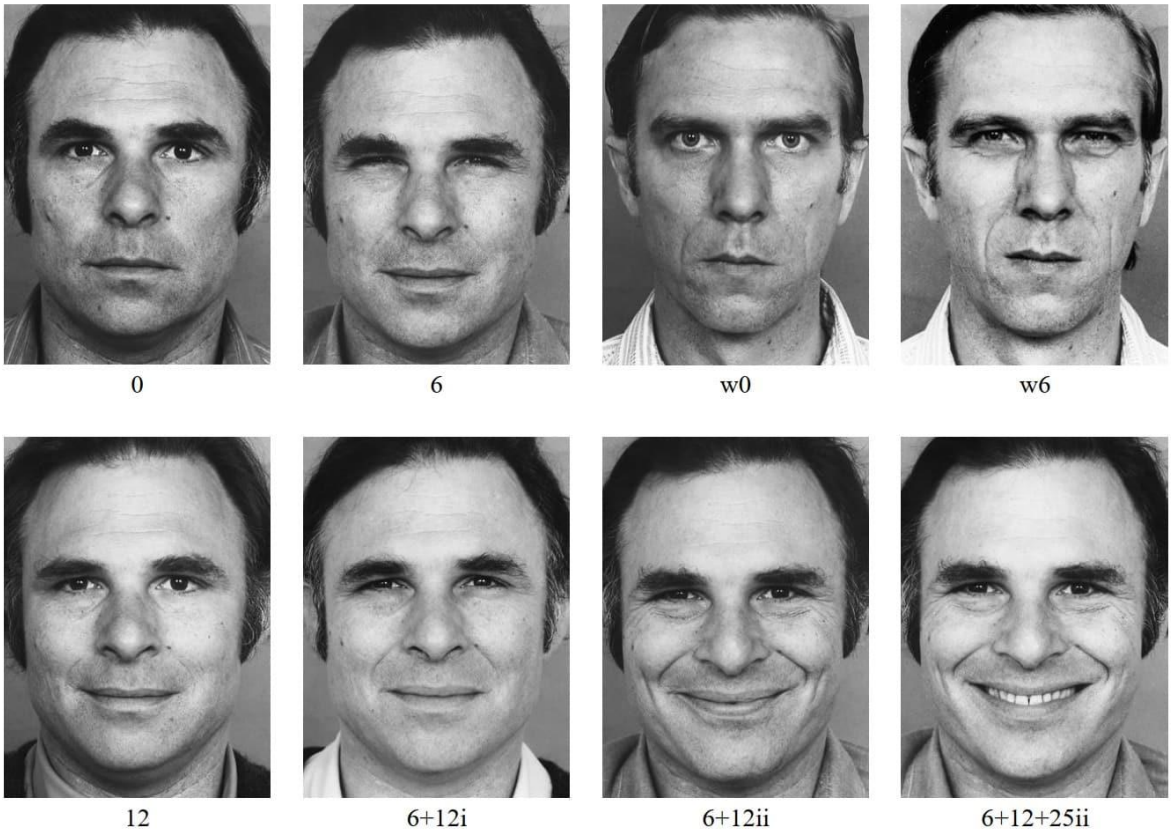


Рисунок 1.2 – Класифікація обличчя чоловіка за системою FACS; як ми можемо бачити, 12 є соціальною посмішкою, а 6+12i та інші варіації вже відображають щасливу посмішку

Саме тому, для цього дослідження розуміння системи FACS необхідне, оскільки сучасні технології комп’ютерного зору та розпізнавання емоцій

працюють так: спочатку визначається набір активних AU, а потім на основі їх комбінації класифікується емоція. Такий підхід робить систему більш стійкою та дає можливість розпізнавати велику кількість різноманітних емоцій [3,5].

1.1.3 Дименсійна модель (Багатовимірний підхід)

Проте в категоріального підходу все ж є свої недоліки, наприклад проблема з точним описом змішаних нестандартних емоцій та переходів між ними. Для вирішення цієї проблеми зазвичай використовують багатовимірний підхід до класифікації, найбільш відомим прикладом якого можна вважати Циркумплексну модель афекту Джеймса Рассела.

В цій моделі емоції розглядаються не дискретна мітка, за якою можна класифікувати, а більше як вектор в неперервному двовимірному просторі. Використовуються таки поняття (див. рис. 1.3):

– валентність (Valence, вісь X): характеризує забарвленість емоції, використовуючи знак, від негативних «неприємних» значень, до позитивних «приємних»;
– збудження (Arousal, вісь Y): характеризує рівень енергетичного стану організму людини – від сонливості й спокою, до активності чи високого рівню напруги.

Як приклад, в цій системі координат:

– радість має позитивну валентність і середнє збудження;
– гнів має негативну валентність та високе збудження;
– депресія має негативну валентність та низьке збудження.

З боку систем комп’ютерного зору використання цієї моделі змінює задачу з класифікації на задачу регресії, що дозволяє більш точно аналізувати складні психоемоційні стани людини [4].

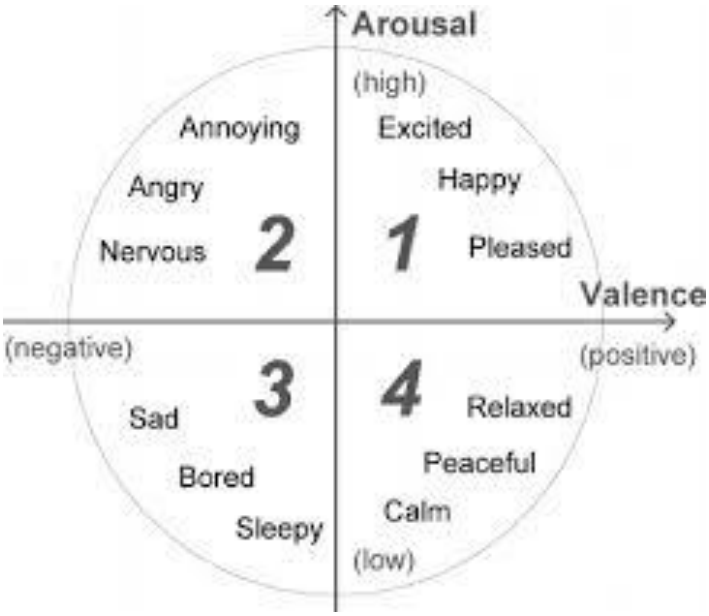


Рисунок 1.3 – Модель афекту Джеймса Рассела

1.1.4 Статичні та динамічні ознаки обличчя

В процесі аналізу алгоритмів комп'ютерного зору дуже важливо розрізняти також і типи візуальної інформації, яку можна отримати від зображення обличчя людини. З боку психофізіології, візуальні ознаки на обличчі можна поділити на такі класи:

- статичні ознаки: характерні риси, які не змінюються або змінюються дуже повільно (наприклад форма черепа, колір шкіри, розташування очей, вікові зморшки, пігментація). У контексті розпізнавання емоцій системами комп'ютерного зору такі риси можуть бути «шумом» і приводити до помилкових спрацьовувань. Наприклад, людина з опущеними від природи куточками очей може бути помилково визначена як «сумна», хоча це і не буде правдою;
- динамічні ознаки: тимчасові зміни в обличчі людини, які виникають тільки в момент якоїсь реакції, здебільшого емоційної.

До динамічних ознак можна віднести:

- появу мімічних зморшок (на лобі, навколо очей, носогубні складки);
- зміну геометрії обличчя (ступінь відкриття очей і рота);

– зміну текстури шкіри (при напрузі).

Сучасні нейронні мережі вже навчаються виділяти саме динамічні патерни та ігнорувати статичні особливості конкретної людини. Такий підхід значно підвищує точність класифікації. Успішність такого навчання безпосередньо залежить від якості навчальної вибірки та архітектури мережі [6].

1.1.5 Постановочні та спонтанні емоції

Ще одним важливим аспектом у системі аналізу емоцій комп'ютерним зором є реалізація визначення відмінностей між постановочними та спонтанними емоціями:

– постановочні емоції це ті вирази обличчя, які людина демонструє спеціально (наприклад, актори чи люди, яких попросили посміхнутися для фото). Такі емоції мають високу інтенсивність, навіть можна сказати гіпертрофованість, і, як правило, легко класифікуються за ідеальними шаблонами Екмана. На таких емоціях зазвичай навчалися старі моделі;
– спонтанні емоції виникають мимоволі, в реальних життєвих ситуаціях. Вони можуть мати низьку інтенсивність, бути асиметричними і перемішуватись з іншими емоціями.

Як показують дослідження, міміка при щирій і фальшивій емоції міміка задіює різні нейронні шляхи у мозку (пірамідальна і екстрапірамідальна системи). Як приклад, щира посмішка (посмішка Дюшенна) обов'язково задіє м'язи навколо очей (Orbicularis oculi), тоді як соціальна посмішка обмежується тільки м'язами рота (Zygomaticus major). Для систем комп'ютерного зору розпізнавання спонтанних емоцій є більш комплексним завданням, і вимагає використання більш досконалих технологій та наборів даних [7].

1.2 Огляд методів комп’ютерного зору для розпізнавання емоцій

Автоматичне розпізнавання емоцій (Facial Emotion Recognition, FER) є однією з класичних задач комп’ютерного зору. Традиційно, можна виділити такі етапи загального алгоритму обробки даних в таких системах:

а) визначення обличчя та попередня обробка: виявлення області обличчя на зображенні, вирівнювання, нормалізація розміру та освітлення;

б) вилучення ознак: перетворення зображення у вектор інформативних характеристик;

в) класифікація: віднесення ознак до однієї з категорій емоцій, використовуючи отриману інформацію (наприклад, за допомогою методу опорних векторів SVM або нейронної мережі).

В залежності від обраного способу вилучення ознак, методи комп’ютерного зору поділяються на традиційні та методи на основі глибокого навчання.

1.2.1 Традиційні методи

До того, як прийшла епоха повсюдного використання глибокого навчання, дослідники використовували алгоритми, що базуються на сконструйованих власноруч ознаках. Вони й зараз актуальні у системах, які мають обмежені обчислювальні ресурси. Такі методи поділяються на два типи:

а) Геометричні методи: вони засновані на відстежуванні координат ключових точок обличчя (наприклад куточків очей, рота, кінчика носа). Вектор ознак формується із відстаней та кутів між цими точками. Перевагами таких методів можна назвати стійкість к змінам в освітленні. До недоліків же можна віднести занадто високу чутливість до точності визначення точок та кута повороту голови.

б) Методи на основі зовнішнього вигляду: вони аналізують текстуру шкіри та градієнти яскравості зображення для виявлення спільних шаблонів (складок, зморшок). Найбільш відомими алгоритмами цього типу є:

– LBP (Local Binary Patterns) або локальні бінарні шаблони. Цей метод кодує текстуру та порівнює яскравість центрального пікселю з сусідніми. LBP показав високу ефективність у ранніх системах FER завдяки своїй обчислювальній простоті та стійкості до монотонних змін яскравості (див. рис. 2.1) [8];
– HOG (Histogram of Oriented Gradients) або гістограми спрямованих градієнтів. Цей метод описує форму та контури обличчя використовуючи розподіл напрямків градієнтів (перепадів яскравості). Спочатку його розробили для виявлення пішоходів, але потім успішно адаптували для аналізу міміки (див. рис. 2.2) [9].

Подальша класифікація отриманих ознак зазвичай виконувалась за допомогою метода опорних векторів (Support Vector Machines, SVM) або випадкового лісу (Random Forest).

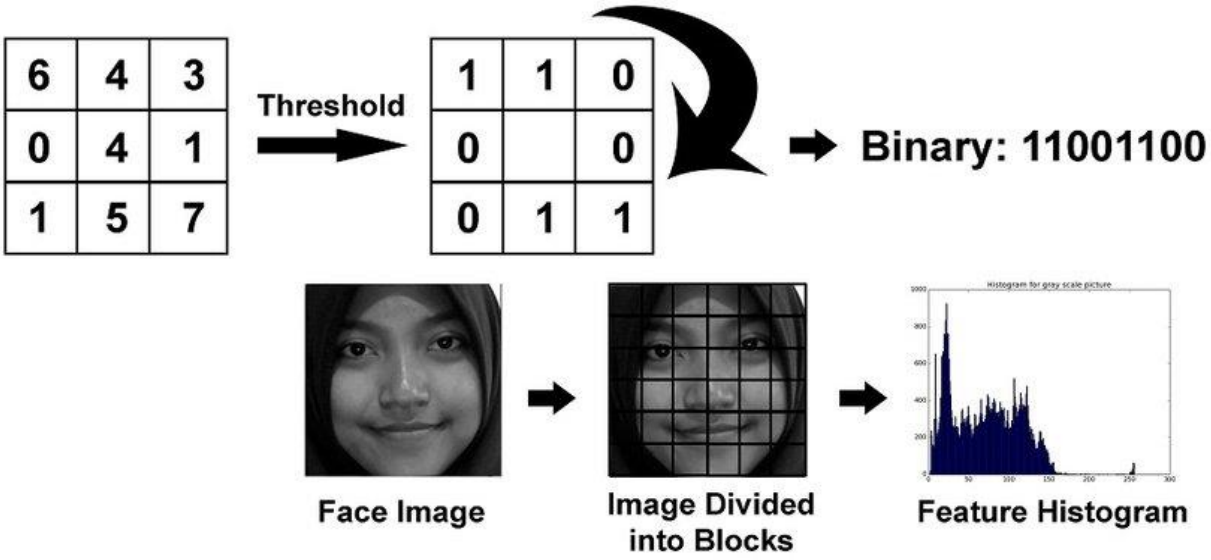


Рисунок 2.1 – Приклад роботи алгоритму локальних бінарних шаблонів (LBP)

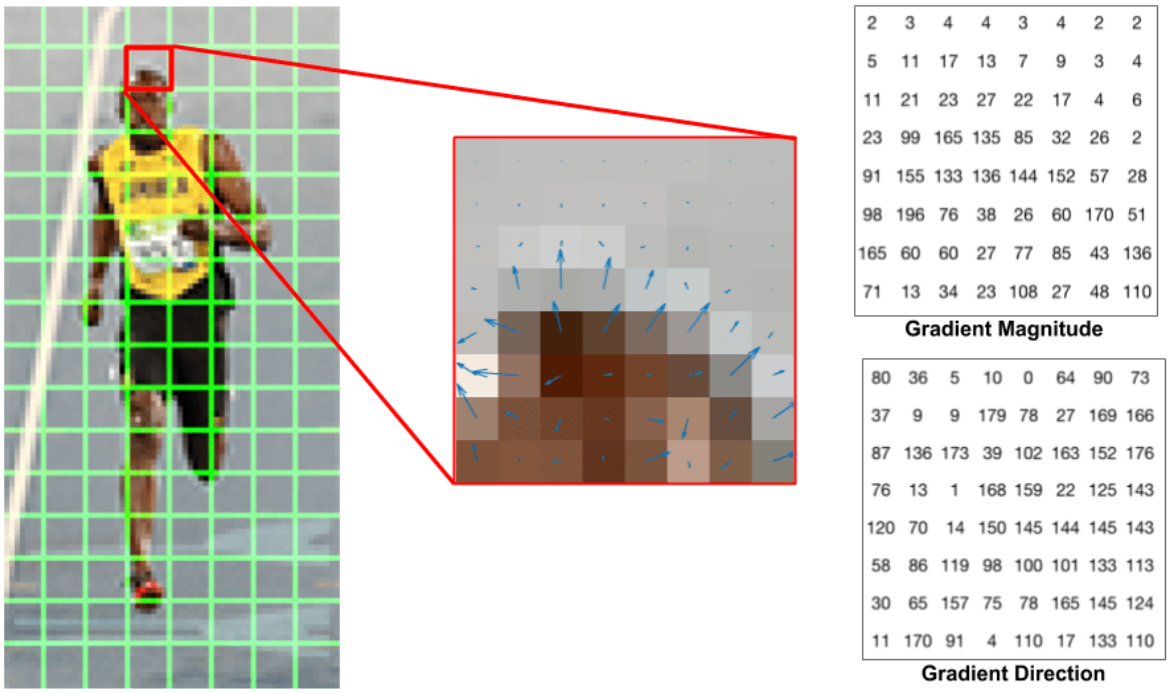


Рисунок 2.2 – Приклад роботи алгоритму гістограми спрямованих градієнтів (HOG)

1.2.2 Методи глибокого навчання (Deep Learning)

Після 2012 року відбувся якісний стрибок у розвитку технологій комп'ютерного зору, здебільшого пов'язаний із впровадження згорткових нейронних мереж (Convolutional Neural Networks, CNN). Основна відмінність CNN від традиційних методів полягає в автоматичному вилученні ознак. Нейронна мережа самостійно навчається виділяти найбільш важливі шаблони в процесі навчання (End-to-End Learning).

Типова архітектура CNN для розпізнавання емоцій є ієрархічною структурою, яка складається з трьох видів шарів:

– згорткові шари (Convolutional Layers) це основа мережі. За допомогою ядер (фільтрів, що навчаються) вони виділяють на зображенні візуальні примітиви. Перші шари реагують на прості елементи, такі як лінії та кути, а більш глибокі формують з них вже складні об'єкти, такі як очі, губи, зморшки;

– шари пулінгу (Pooling Layers) виконують операцію вибірки карт ознак для зниження їхньої розмірності. Таким чином зменшується обчислювальне навантаження, а модель стає стійкою до невеликих зміщень обличчя у кадрі;
– повнозв’язні шари (Fully Connected Layers) розташовані в кінці мережі та виконують кінцеву класифікацію, зіставляючи виділені високорівневі ознаки з конкретними класами емоцій.

Для задач класифікації емоцій в якості функції активації прихованих шарів зазвичай використовується ReLU (Rectified Linear Unit) для прискорення навчання, а у вихідному шарі – функція Softmax, яка перетворює результат роботи нейронної мережі в розподіл ймовірностей (наприклад «Радість – 85%», «нейтрально – 15%») [10].

Хоча методи глибокого навчання мають високу точність, вони також вимагають значних ресурсів і великих обсягів розмічених даних для запобігання перенавчанню.

1.2.3 Огляд ключових архітектур для порівняльного аналізу

Можна навести такі приклади:

– VGG (Visual Geometry Group): ця архітектура була запропонована Оксфордським університетом. Характерною рисою є використання блоків із простих згортків 3x3. Хоча ця архітектура є дуже простою та зрозумілою, і має високу точність, вона також має дуже велику кількість параметрів (близько 138 мільйонів), що робить її важкою для зберігання і повільною для інференсу на CPU [11];
– ResNet (Residual Networks): революційна архітектура від Microsoft, яка вирішує проблему загасання градієнта при навчанні дуже глибоких мереж за допомогою залишкових зв’язків (skip connections). Ця архітектура дозволяє тренувати мережі глибиною 50, 101 і 152 шари.

При тому вона тримає гарний баланс між точністю і швидкістю, тому є базовою моделлю для більшості сучасних систем розпізнавання [12];

– MobileNet і EfficientNet (Легковагові моделі): ці архітектури розроблені Google спеціально для мобільних пристроїв і вбудованих систем. Через те, що вони використовують згортки, розділені за глибиною (Depth-wise Separable Convolutions), кількість обчислюваних операцій скорочується у 8 – 9 разів у порівнянні зі стандартними згортками при мінімальній втраті точності. Ці моделі ідеально підходять для порівняння продуктивності на пристроях без GPU [13].

1.2.4 Перенос навчання

Однією із головних проблем навчання моделей для розпізнавання емоцій є обмежений розмір розмічених наборів даних. Для вирішення цієї проблеми використовується метод переносу навчання (або Transfer Learning): модель попередньо навчається на величезному наборі даних загального призначення (наприклад, ImageNet або VGGFace2 для розпізнавання облич), а потім її останні шари донавчаються (fine-tuning) на спеціальній задачі класифікації емоцій. Такий підхід значно прискорює збіжність мережі та підвищує точність [14].

1.3 Огляд існуючих наборів даних (Datasets)

Як ми вже казали раніше, на точність роботи систем на основі глибокого навчання сильно впливають набори даних, на яких навчались ці системи. В галузі комп’ютерного зору набори даних можна поділити на такі групи: лабораторні (posed) та природні (in-the-wild).

1.3.1 Лабораторні набори даних

Такі набори даних називаються лабораторними, бо вони створювалися в повністю контрольованих, «лабораторних» умовах. Наприклад, були запрошені актори, яких просили зобразити на обличчі якусь емоцію, і все це у фіксованому ракурсі та з використанням високоякісної апаратури для зйомок й освітлення.

Найвідомішим лабораторним набором даних можна назвати CK+ (The Extended Cohn-Kanade Dataset): він є найбільш цитованим стандартом і використовується у великій кількості базових досліджень. Містить 593 відеопослідовності від 123 суб'єктів. Кожне відео починається з нейтрального стану обличчя, а закінчується піком емоції. Усі зображення розмічені відповідно до системи FACS і класифіковані за 7 базовими емоціями Екмана.

Проте, цей набор має і свій недолік: через те, що емоції є надто «ідеальними», гіпертрофованими, цей набор даних погано відображає реальні умови експлуатації систем [15].

1.3.2 Природні набори даних

Для того, щоб система могла працювати в умовах реального світу, потрібні датасети, які усі ці умови передають: різний рівень освітлення, різний кут нахилу голови, якісь речі, що перекривають зображення (окуляри, руки біля обличчя), низька якість зйомки. Із таких наборів даних можна виділити:

– FER-2013: один із самих відомих наборів, який було представлено на змаганні Kaggle в 2013 році. Він містить 35887 зображень розміром 48x48 пікселів у градаціях сірого. Ці зображення були відібрані автоматично через пошукову систему Google за ключовими словами, пов'язаними з емоціями. Низька роздільна здатність та наявність

шуму є особливістю цього набору даних, бо він ідеально відтворює реалістичні умови і є серйозним бенчмарком для нейромереж. Саме на цьому наборі навчається велика кількість легковажних моделей [16];

– AffectNet: в наш час це найбільша база даних виразів обличчя людини, вона містить більше 1 мільйона зображень, які були зібрані в інтернеті. Особливістю цього набору даних можна назвати те, що в ньому дані розмічені як по категоріальному методу (8 емоцій), так і по діменсійному (осі Valence і Arousal). Саме тому моделі, навчені на цьому наборі, можуть розпізнавати найтонші емоційні відтінки [17].

1.4 Аналіз готових рішень та бібліотек

Зараз існує вже велика кількість готових рішень, високорівневих бібліотек та API, які надають доступ до вже навчених моделей (pre-trained).

1.4.1 Open-Source бібліотеки (Python)

Наявні рішення:

– DeepFace: легкий фреймворк для аналізу ознак обличчя на Python. Він працює як зовнішній шар (wrapper) для таких моделей розпізнавання лиць як VGG-Face, Google FaceNet, OpenFace. Цей фреймворк використовує власну нейронну мережу, яка була навчена на FER-2013. Цей фреймворк легко інтегрувати, він підтримує такі відомі бібліотеки як TensorFlow та PyTorch [20];

– FER (Facial Expression Recognition): спеціально розроблена для класифікації емоцій бібліотека. Працює в два етапи: детектор MTCNN шукає обличчя, а мережа на архітектурі VGG класифікує емоції. Ця бібліотека оптимізована для роботи з відеопотоком і має інструменти для візуалізації результатів у реальному часі.

1.4.2 Хмарні сервіси (Cloud API)

Як альтернатива локальним моделям виступають хмарні рішення від великих корпорацій. Вони надають доступ до моделей через REST API.

– Azure Face API (Microsoft): дуже ефективний сервіс, який на запит повертає координати обличчя та ймовірності 8 стандартних емоцій на зображенні. Має високу точність навіть при роботі зі складними зображеннями (повороти обличчя, погане освітлення), але потребує гарного стабільного інтернет з’єднання та є платним сервісом (хоча є безкоштовний тестовий варіант);
– Google Cloud Vision API: надає функцію faceDetection, яка оцінює ймовірність наявності чотирьох емоцій на зображенні: радість, сум, гнів і здивування. Ймовірність задана показником від «Very Unlikely» до «Very Likely».

2 ПРОЕКТУВАННЯ СИСТЕМИ ТА МЕТОДОЛОГІЯ ПОРІВНЯЛЬНОГО АНАЛІЗУ

В цьому розділі буде обґрунтовано вибір архітектур нейронних мереж для порівняльного аналізу, описана конфігурація апаратно-програмного комплексу та буде сформований алгоритм проведення тестування на базі наявних обчислювальних ресурсів.

2.1 Обґрунтування вибору технологій для дослідження

Для проведення порівняльного аналізу були обрані три архітектури, кожна з яких представляє різний підхід до створення систем комп'ютерного зору. Основною складністю при виборі технологій була необхідність знайти баланс між точністю розпізнавання та обчислювальною складністю при роботі на CPU. Особливості програмно-апаратної реалізації, що зумовили вибір даних архітектур для тестування саме на CPU, детально описані в підрозділі 2.2.

2.1.1 Каскадний підхід – бібліотека FER (MTCNN)

Ця бібліотека реалізує класичний конвеєрний підхід до аналізу обличчя та визначення емоцій: для детекції використовується каскадна мережа MTCNN (Multi-task Cascaded Convolutional Networks), яка працює за принципом «від грубого до точного» – послідовно відсіює фонові області. Подібні класичні методи детекції описані в фундаментальних роботах з аналізу облич [6].

Через те, що ця система має каскадну структуру, вона споживає мінімум ресурсів якщо в кадрі немає облич, або є який шум, який перешкоджає нормальному аналізу, та може ефективно використовувати багатопотоковість

процесору. В цьому дослідженні ця технологія буде розглядатись як «базовий рівень» для оцінки приросту продуктивності інших, більш складних моделей.

2.1.2 Глибокі мережі – DeepFace (VGG-Face)

Ця архітектура є представником більш важких та високоточних моделей. Вона базується на VGG-16 і є інтегрованою в фреймворк DeepFace [18].

Як було описано в роботі [11], головним принципом VGG є відмова від використання фільтрів великого розміру та перехід до послідовного застосування фільтрів малого розміру (3 x 3).

Обґрунтування з боку математики: стек з двох шарів (3 x 3) ефективне рецептивне поле (receptive field), яке еквівалентне одному шару (5 x 5), але містить меншу кількість параметрів та вносить більше нелінійності, що позитивно впливає на ефективність мережі та її здатність виділяти складні ознаки.

Проте, хоча ця архітектура має високу точність, повнозв’язні шари VGG містять величезну кількість параметрів (більше 130 мільйонів), і це створює високе навантаження на пам’ять і кеш процесора.

2.1.3 Ефективна архітектура – MobileNet

Ця архітектура спеціально розроблена для роботи з обмеженими обчислювальними ресурсами, наприклад на мобільних пристроях або вбудованих системах [13].

Ключовою особливістю архітектури MobileNet, що відрізняє її від стандартних CNN є використання глибинно-роздільних згорток (Depthwise Separable Convolutions). Звичайна згортка одночасно фільтрує та комбінує канали, в той час як MobileNet розділяє цей процес:

– Depthwise convolution: один фільтр застосовується до кожного вхідного каналу окремо;

– Pointwise convolution: для лінійної комбінації виходів глибинної згортки використовується згортка (1 x 1).

Таким чином, в 8–9 разів скорочується кількість обчислень і параметрів моделі, що забезпечує високу ефективність на CPU без істотної втрати точності.

2.2 Характеристика програмно-апаратного комплексу дослідження

Далі будуть описані технічні характеристики апаратного забезпечення та середовища розробки, які будуть використані для забезпечення наочності результатів дослідження.

2.2.1 Апаратне забезпечення

Вибір компонентів зумовлений необхідністю перевірки ефективності алгоритмів на обладнанні споживчого класу (Consumer Grade) без використання серверних прискорювачів.

Варто зазначити, що сучасним стандартом для тренування нейронних мереж в промислових обсягах є використання GPU компанії NVIDIA з архітектурою CUDA: їх особливістю є масивний паралелізм, який забезпечує значний приріст продуктивності. Однак, залежність від технології CUDA та висока вартість такого апаратного забезпечення суттєво обмежують коло пристроїв, на яких можуть бути розгорнуті такі системи (проблема Hardware Dependency). Більшість персональних комп’ютерів, ноутбуків або вбудованих систем (IoT) не оснащені потужними дискретними відеокартами, тому однією з додаткових цілей цього дослідження є перевірка можливості ефективного використання систем розпізнавання емоцій при використанні виключно обчислювальних потужностей CPU. Це дозволить довести можливість впровадження алгоритмів на широкому спектрі обладнання без додаткових апаратних витрат.

Актуальні параметри:

– центральний процесор AMD Ryzen 7 2700X: має 8 ядер та 16 фізичних потоків. Він підтримує набір інструкцій AVX2 (Advanced Vector Extensions), що прискорює тензорні обчислення на CPU;
– Оперативна пам’ять (RAM): 16 ГБ DDR4, такий обсяг дозволяє утримувати в буфері весь тестовий датасет та ваги нейронних мереж, що нівелює затримки на операції з диском.

2.2.2 Системне та інструментальне програмне забезпечення

В якості програмної платформи обрано стек технологій, який відповідає сучасним стандартам у галузі машинного навчання:

– операційна система: Windows 10. Через те, що ця ОС є найпоширенішою у світі, це дозволяє відтворити наближені до реального робочого місця користувача умови;
– мова програмування: Python. Ця мова має розвинену екосистему бібліотек для роботи з технологією комп’ютерного зору. Також, механізм віртуальних середовищ (venv) мінімізує можливість помилок, пов’язаних з конфліктами версій;
– середовище розробки: Visual Studio Code/ PyCharm. Найпопулярніші IDE для комфортного написання коду та дебагінга;
– система контролю версій: Git – як еталон таких систем, найкращий вибір для цього дослідження.

2.2.3 Бібліотеки машинного навчання та обробки даних

Основу програмної реалізації складають спеціалізовані бібліотеки, які можна розділити на такі категорії:

а) модуль комп’ютерного зору OpenCV (Open Source Computer Vision Library): використовується для низькорівневої роботи з зображеннями.

Основними функціями є декодування графічних файлів, конвертація кольору, білінійна інтерполяція при зміні розміру кадру та візуалізація результатів;

б) модуль обчислень та нейронних мереж:
   – TensorFlow: високопродуктивна платформа для чисельних обчислень. Так як дослідження орієнтується на роботу з CPU, буде використана CPU-оптимізована збірка, яка задіє мультипотоковість та векторні інструкції процесора;
   – Keras: інтерфейс для роботи з TensorFlow, він дозволяє швидко завантажувати архітектури та маніпулювати шарами;
   – DeepFace: гібридний фреймворк, що надає єдиний інтерфейс для роботи з моделями VGG-Face, Google FaceNet тощо;

в) модуль аналізу та метрик:
   – NumPy: фундаментальна бібліотека для реалізації лінійної алгебри. Усі зображення в пам’яті комп’ютера представлені як багатовимірні масиви (тензори) NumPy у форматі даних float32. Над якими виконуються матричні операції;
   – Pandas: бібліотека для структурованого зберігання результатів експериментів (DataFrames), їх фільтрації та експорту у формат CSV для подальшого аналізу;
   – Scikit-learn: бібліотека що забезпечує реалізацію розрахунків метрик якості (Precision, Recall, F1-score) та побудови матриці плутанини.

2.3 Методика підготовки даних та алгоритм тестування

Ефективність роботи систем машинного навчання сильно залежить від якості навчальної вибірки та коректності підготовки вхідних даних. В цьому дослідженні використовується стандартизований процес обробки, що

забезпечує приведення різних візуальних даних до єдиного тензорного формату для вхідних шарів нейронних мереж.

2.3.1 Характеристика набору даних (FER-2013)

Для проведення порівняльного експерименту буде використовуватись набір даних FER-2013 (Facial Expression Recognition 2013 Dataset), який було представлено на конференції ICML [16]. Цей набір було обрано через те, що він має переваги «складного» для обробки набору даних: зібрані «in-the-wild» фотографії, низьку роздільну здатність зображень, різні кути повороту голови та часткові перекриття обличчя. Це дозволяє перевірити стійкість алгоритмів до умов схожих з реальними.

Для тестування буде використовуватись окрема частина набору даних, яка називається Test Split. Вона не використовується під час навчання моделей.

Структура тестової вибірки:

– загальний об’єм: 7178 зображень;
– формат: одноканальні зображення (Grayscale) розміром 48 x 48 пікселів;
– незбалансований розподіл класів: 1758 зображень категорії Happy, 110 зображень категорії Disgust. Це вимагає використання метрики F1-score замість простої Accuracy для об’єктивної оцінки [21].

2.3.2 Алгоритм попередньої обробки (Preprocessing Pipeline)

Процес обробки, через який проходять сирі дані перед тим як попасти до нейронної мережі, представлений наступним чином:

а) перетворення колірного простору (Grayscale Conversion): більшість архітектур для аналізу емоцій працюють з інтенсивністю світла замість колірної інформації, від якої позбавляються. Це перетворення

RGB зображення в одноканальне Y виконується за стандартом ITU-R 601-2:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B, \quad (2.1)$$

де R, G, B – значення каналів червоного, зеленого та синього кольорів відповідно [19];

б) масштабування (Resizing & Interpolation): різні архітектури вимагають різні розмірності вхідного тензора:
   – для Custom CNN (MobileNet) зображення масштабуються до 48 x 48 (або 224 x 224 в залежності від реалізації Transfer Learning);
   – для DeepFace (VGG-Face) потрібен розмір 224 x 224. При зміні розміру використовується білінійна інтерполяція (Bilinear Interpolation) – це забезпечує оптимальний баланс між швидкістю обчислень та збереженням деталізації на низькій роздільний здатності, прибираючи артефакти пікселізації;

в) нормалізація даних (Normalization): значення яскравості пікселів, що лежать в діапазоні [0, 255], зводяться до діапазону [0, 1]:

$$X_{norm}^{(i,j)} = \frac{X_{raw}^{(i,j)}}{255.0}, \quad (2.2)$$

де X_{norm} – це зведені значення яскравості; X_{raw} – оригінальні значення яскравості.

Це необхідно для прискорення збіжності градієнтного спуску і запобігання насичення функцій активації (vanishing gradients) в глибоких шарах мережі [20];

г) формування пакета (Batching): для ефективного використання кешу процесора, оброблені зображення об’єднуються в пакети (батчі).

Додається четвертий вимір, після чого тензор набуває форми (BatchSize, Height, Width, Channels).

2.3.3 Протокол тестування

Для забезпечення відтворюваності результатів (reproducibility) встановлено такий алгоритм тестування:

а) ініціалізація середовища: для мінімізації впливу на результати виміру часу інференсу відключаються всі фонові процеси ОС (які можливо). Завантаження ваг моделей в оперативну пам’ять відбувається до початку таймера;

б) цикл інференсу: проходження по тестовому набору даних здійснюється послідовно (iteration). Для кращого результату дані не перемішуються;

в) фіксація метрик: вимірюється час обробки кожного батча (час, витрачений на завантаження зображення з диска не враховується). Враховується тільки час роботи методу .predict(). Потім, вектор ймовірностей (Softmax output) перетворюється в мітку класу за допомогою функції:

$$\hat{y} = \text{argmax}(p), \tag{2.3}$$

де p – це вектор ймовірностей, передбачений мережею;

г) порівняльний аналіз: отримані прогнози $\hat{y}$ зберігаються в єдиному DataFrame разом із справжніми мітками y_{true} для подальшого розрахунку матриці помилок (Confusion Matrix).

2.4 Система метрик для оцінки ефективності алгоритмів

Для того, щоб якісно та комплексно провести порівняльний аналіз, треба окрім оцінки здатності нейронної мережі коректно класифікувати емоції ще оцінити ресурси, які витрачаються на цей процес. Через це метрики поділяються на дві групи: метрики якості розпізнавання та метрики обчислювальної ефективності.

2.4.1 Метрики якості розпізнавання

Основою для розрахунку всіх статистичних показників є матриця помилок (Confusion Matrix). Для задачі класифікації емоцій (на 7 емоцій) створюється квадратна матриця розмірністю (7 x 7), у якій елемент C_{ij} показує кількість зображень, що дійсно належать до класу i, але передбачені алгоритмом як клас j. На основі елементів матриці розраховуються такі показники [21]:

– точність (Accuracy): базова метрика, яка показує частку правильних відповідей щодо загальної кількості прикладів:

$$Accuracy = \frac{\sum_{i=1}^{K} C_{ii}}{N}, \tag{2.4}$$

де C_{ii} – діагональні елементи матриці (правильні передбачення); N – загальний обсяг тестової вибірки; K – кількість класів (7 емоцій);

– презиційність (Precision) та повнота (Recall): оскільки набор даних FER-2013 є незбалансованим, Accuracy може показувати некоректні результати. Тому для кожного класу i розраховуються окремі показники. Презиційність для класу i (Precision) показує, скільки із зображень, які були помічені системою, наприклад, як «страх», дійсно були «страхом»:

$$Precision_i = \frac{TP_i}{TP_i + FN_i}, \quad (2.5)$$

а повнота для класу i (Recall) показує, яку частину реального «страху» система змогла знайти:

$$Recall_i = \frac{TP_i}{TP_i + FN_i}, \quad (2.6)$$

де TP (True Positive) – істинно позитивні спрацьовування системи;

FP (False Positive) – помилкові спрацьовування (шум);

FN (False Negative) – пропуски цілі;

– F1-Score (Macro-averaged): інтегральна метрика, що є гармонійним поєднанням метрик точності та повноти. Для отримання єдиної оцінки роботи системи використовується макро-усереднення (середнє арифметичне F1-метрик всіх класів). Це дозволяє уникнути домінування мажоритарних класів (таких як «Happy») над міноритарними (такими як «Disgust») в підсумковій оцінці:

$$F1_{macro} = \frac{1}{K}\sum_{i=1}^{K} 2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i}, \quad (2.7)$$

де $F1_{macro}$ – макро-усереднене значення F1-метрики;

K – загальна кількість класів емоцій (у рамках даного дослідження K = 7);

$Precision_i$ – значення точності для i-го класу;

$Recall_i$ – значення повноти для i-го класу;

i – індекс поточного класу.

2.4.2 Метрики обчислювальної ефективності

Враховуючи те, що це дослідження розраховано на перевірку ефективності алгоритмів на Edge-системах на базі CPU, критично важливими є показники часових характеристик [22]:

– час інференсу (Inference Latency): час, який витрачає модель на повний цикл обробки одного зображення (Forward Pass), не враховуючи час на завантаження даних з диска:

$$t_{infer} = t_{end} - t_{start}, \tag{2.8}$$

де t_{infer} – час інференсу;

t_{end} – час завершення інференсу;

t_{start} – час початку інференсу.

Цей показник вимірюється в мілісекундах (мс), розраховується по всьому тестовому набору для досягнення статистичної достовірності;

– кадри за секунду (FPS – Frames Per Second): кількість кадрів, яку система здатна обробити за одну секунду. Ця метрика є похідною від часу інференсу і показує, чи зможе система працювати в реальному часі:

$$FPS = \frac{1000}{T_{avg}}, \tag{2.9}$$

де T_{avg} – середня кількість часу, яку витрачає система на обробку одного кадру в мс.

Для комфортної взаємодії людини з системою (HCI – Human Computer Interaction) значення FPS має бути не нижче за 24-30 кадрів/с. Якщо FPS менше 10, система вважається непридатною для аналізу мікровиразів обличчя в реальному часі.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЕМОЦІЙ

3.1 Підготовка та налаштування моделей

В цьому проекті для реалізації функціоналу розпізнавання емоцій та для порівняльного аналізу результатів використовується декілька архітектур, кожна з яких має свої особливості при підготовці. Обраними архітектурами є FER, DeepFace та MobileNetV2.

3.1.1 Готові моделі (Out-of-the-box)

Велика кількість моделей (наприклад такі, як розглянуті FER та DeepFace) одразу мають ваги і вже навчені на великих наборах даних. Тому для їх інтеграції в проект не потрібне окреме навчання, просто встановити потрібні бібліотеки та ініціювати коректні ваги стосовно документації. Таким чином, ці моделі готові до інференсу одразу після розгортки середи.

3.1.2 Підготовка моделі MobileNetV2

Стандартна версія цієї мережі (з Keras Applications) навчена на датасеті ImageNet (1000 класів об’єктів), тому не пристосована до розпізнавання виразів обличчя одразу, на відміну від готових моделей. Для того, щоб використовувати архітектуру MobileNetV2 для задачі класифікації емоцій (на 7 класів), її треба попередньо навчити. Для цього була обрана така стратегія: Transfer Learning (навчання з переносом знань) та після нього Fine-Tuning (тонке налаштування). В якості набору даних для навчання було використано

набір FER-2013. Повний лістинг з кодом додаткової програми, яка відповідає за підготовку архітектури та процес навчання, наведено в Додатку А.

Процес навчання був поділений на декілька логічних етапів, ефективність яких можна побачити на отриманих метриках:

а) навчання класифікатора (Feature Extraction): на цьому етапі шари базової моделі (з вагами ImageNet) були заморожені, і навчалась тільки додана повнозв'язкова «голова» мережі. Такий підхід дозволяє адаптувати ваги вихідного шару до нових класів, не відкидаючи ознаки, вивчені базовою мережею.

```
Epoch 14/15
898/898 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step - accuracy: 0.4721 - loss: 1.3789
Epoch 14: val_accuracy improved from 0.46085 to 0.46141, saving model to mobilenet_fer2013_head.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. W
898/898 ━━━━━━━━━━━━━━━━━━━━ 111s 124ms/step - accuracy: 0.4742 - loss: 1.3763 - val_accuracy: 0.4614 - val_loss: 1.4026 - learning_rate: 2.5000e-04
Epoch 15/15
898/898 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step - accuracy: 0.4738 - loss: 1.3780
Epoch 15: val_accuracy did not improve from 0.46141
898/898 ━━━━━━━━━━━━━━━━━━━━ 113s 126ms/step - accuracy: 0.4772 - loss: 1.3721 - val_accuracy: 0.4606 - val_loss: 1.3984 - learning_rate: 2.5000e-04

Результати Етапу 1:
   Найкраща точність валідації: 0.4614
   Фінальна точність валідації: 0.4606

ЕТАП 2: Fine-Tuning...
   Кількість шарів базової моделі: 154
   Буде навчатись шарів: 54 (починаючи з 100-го)

   Learning Rate для fine-tuning: 5e-5 (з автоматичним зниженням)
```

Рисунок 3.1 – Показники першого етапу навчання (заморожені шари)

Як можна побачити із наведених результатів (див. рис. 3.1), модель досягла точності валідації 0.4614, що є базовим рівнем, який вказує на те, що класифікатор почав розрізняти емоції , але узагальнююча здатність моделі все ж таки обмежена вагами, які розраховані на об'єкти ImageNet, а не під обличчя;

б) тонке налаштування (Fine-Tuning): на цьому етапі було проведено «розморожування» глибоких шарів базової моделі (починаючи з 100-го шару, розморожування більшої кількості шарів призвело б до нестабільності) і запуск повторного навчання зі зменшеним кроком (Learning Rate 5e-5). Таким чином, нейронна мережа перебудовує фільтри згортки спеціально для виділення мімічних ознак, таких як вигин брів, форма губ та інше.

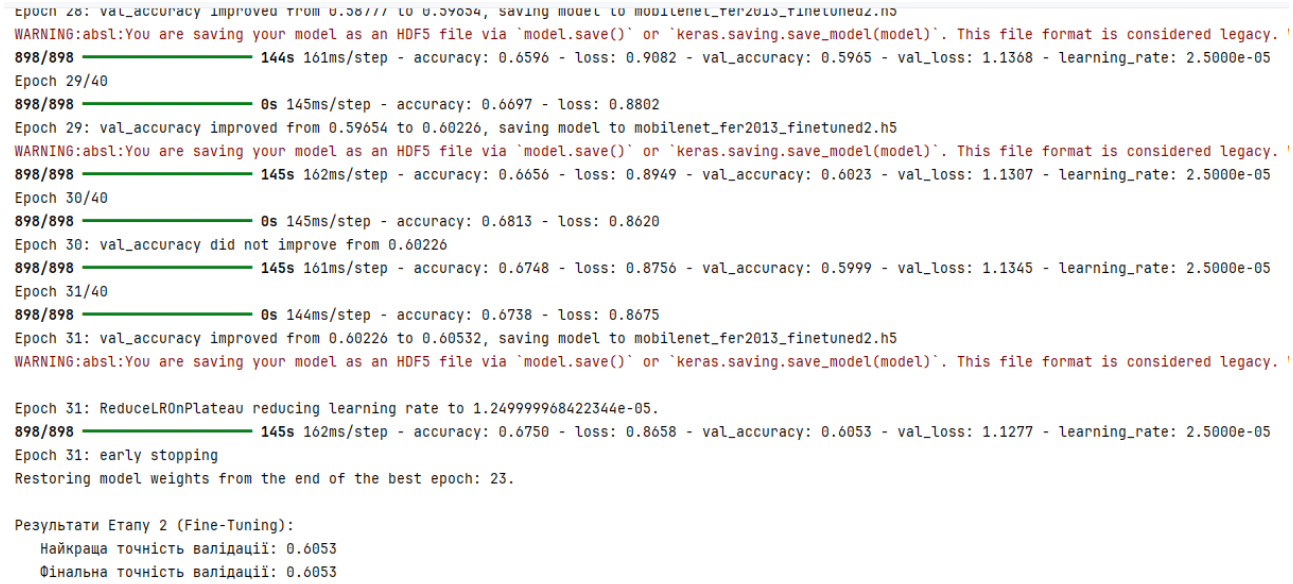


Рисунок 3.2 – Фінальні показники навчання після Fine-Tuning і спрацьовування Early Stopping

Згідно з отриманими результатами (див. рис. 3.2), стратегія тонкого налаштування (Fine-Tuning) дала значний приріст якості. Система раннього зупинення (Early Stopping) припинила навчання на 31-й епосі, щоб не допустити перенавчання (overfitting).

Приріст точності, порівняно з попереднім етапом, склав більше 14%, а фінальна точність на валідації склала 0.6053 (60.53%). Такий результат підтверджує, що «донавчання» глибоких шарів є критично важливим кроком для адаптації MobileNetV2 до задачі розпізнавання виразів обличчя (Facial Expression Recognition).

Програмний засіб виводить графік процесу навчання моделі для кращого розуміння результатів навчання (див. рис. 3.3).

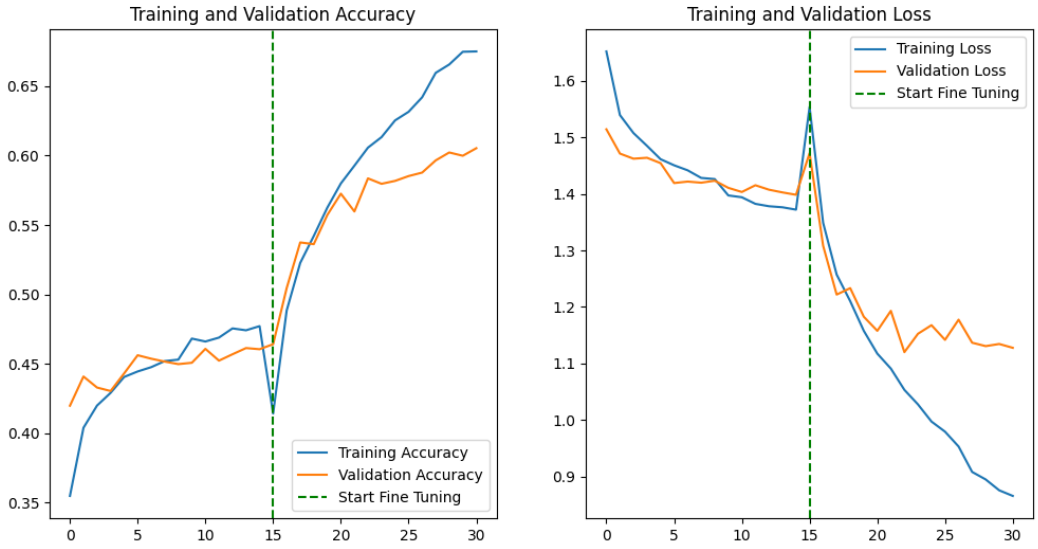


Рисунок 3.3 – Графік останнього етапу навчання, на якому видно, що почалось різке збільшення різниці між тренувальними параметрами та параметрами валідації, що вказує на ризик перенавчання (overfitting)

3.1.3 Розробка програмного модуля для інференсу навченої моделі

Для верифікації результатів навчання та перевірки, чи працює взагалі модель MobileNetV2 як треба, перевірки на реальних, а не тестових, даних, був розроблений окремий скрипт на мові Python. В його функціонал входить виконання інференсу та емуляція роботи підсистеми розпізнавання в кінцевому додатку. Повний лістинг програмного коду скрипту наведено в Додатку Б.

Розроблений програмний модуль працює за таким алгоритмом:

а) ініціалізація моделі: скрипт завантажує архітектуру та ваги, які були отримані на етапі тонкого налаштування за допомогою навчального скрипту, ваги є файлом у форматі .h5 (Hierarchical Data Format);

б) завантаження та попередня обробка зображення: вхідне зображення користувача зчитується за допомогою бібліотеки OpenCV. Модель MobileNetV2 має жорсткі вимоги до розмірності вхідних

даних, тому для зображень реалізуються такі кроки попередньої обробки:

– конвертація зображення з формату BGR (стандарт OpenCV) в RGB (стандарт Keras);
– зміна розміру зображення до 96 х 96 пікселів. Це критично важливий етап, оскільки спроба подати зображення іншої роздільної здатності (наприклад, стандартних 224 х 224) призведе до помилки невідповідності розмірностей тензорів;
– приведення значень пікселів до діапазону [-1, 1] з використанням функції preprocess_input, що є препроцесингом, як той, що був використаний при попередньому навчанні ваг ImageNet;

в) класифікація та інтерпретація результатів: як результат роботи моделі, отримується вектор із 7 чисел (вихід функції активації Softmax), сума яких дорівнює 1. Програмний модуль оброблює цей вектор наступним чином:
– визначається клас з максимальним значенням (домінуюча емоція);
– декодуються індекси (0-6) в зрозумілі для людини назви емоцій (Злість, Радість, Відраза тощо);
– виводиться розподіл впевненості моделі по всіх класах. Це дозволяє аналізувати «граничні», «змішані» стани емоцій (наприклад, коли ймовірність «Радості» 45%, а «Здивування» 40%).

Програмний модуль був протестований на вибірці користувацьких зображень (див. рис. 3.4 та рис. 3.5). Результати підтвердили, що навчена модель коректно завантажується, не конфліктує з версіями бібліотек та успішно виконує свою роль – класифікує емоції на зображеннях, які не брали участі в навчальній вибірці. Таким чином, модель MobileNetV2 була підготовлена до інтеграції в основну програму для порівняльного аналізу.

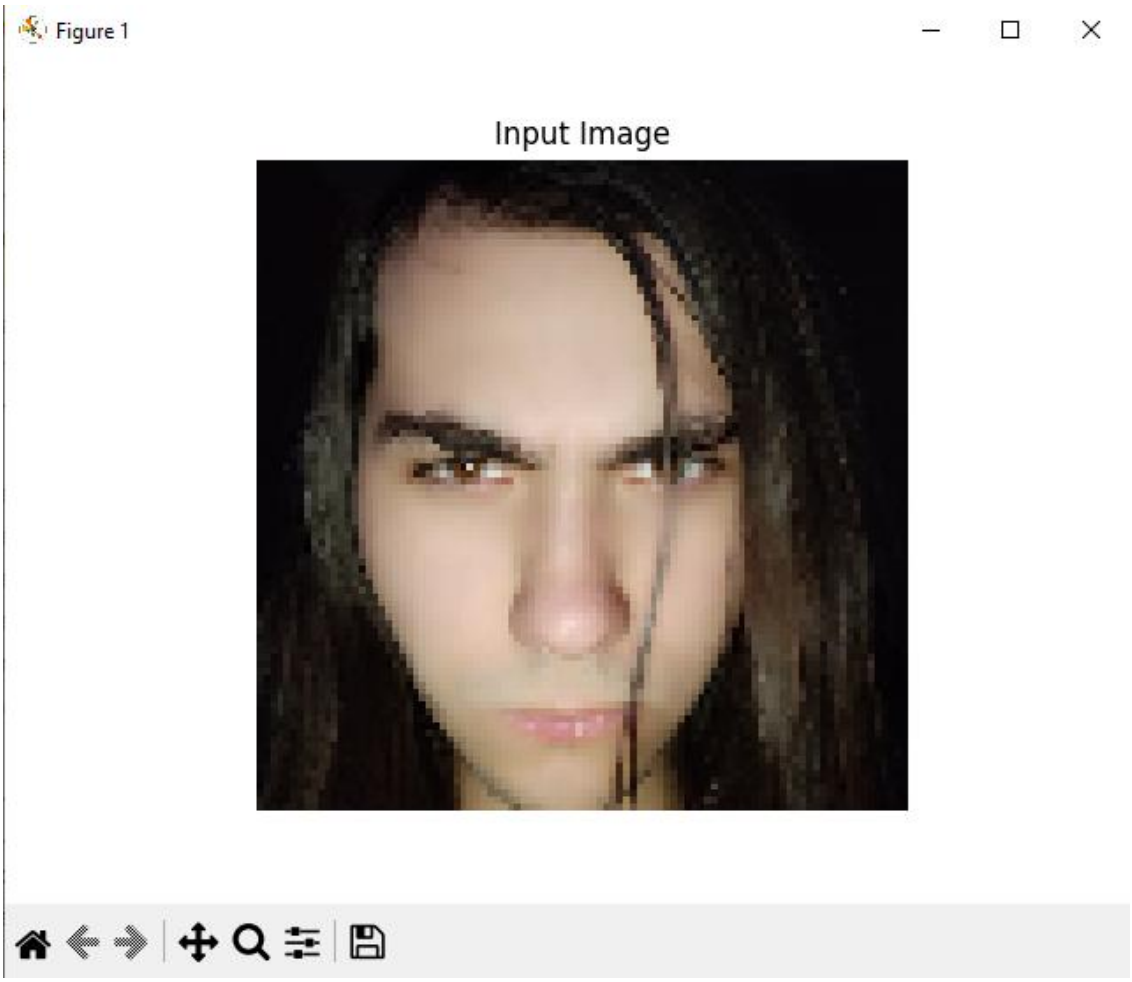


Рисунок 3.4 – Показ обробленого зображення для розпізнавання

```
========================================
РЕЗУЛЬТАТ: Angry (Злість)
Впевненість: 84.37%
========================================

Детальний розподіл ймовірностей:
----------------------------------------
Angry (Злість)            : 0.8437 (84.37%)
Disgust (Відраза)         : 0.0001 (0.01%)
Fear (Страх)              : 0.0216 (2.16%)
Happy (Радість)           : 0.0005 (0.05%)
Sad (Сум)                 : 0.0056 (0.56%)
Surprise (Здивування)     : 0.1285 (12.85%)
Neutral (Нейтрально)      : 0.0000 (0.00%)
----------------------------------------
```

Рисунок 3.5 – Консольний вивід результатів роботи програмного модуля

3.2 Розробка програмного засобу для порівняльного аналізу

Для об’єктивної оцінки ефективності різних моделей було створено спеціальний програмний засіб (бенчмарк). Основною метою програми є проведення порівняльного тестування трьох різних архітектур для розпізнавання емоцій на однаковому наборі даних та в однакових умовах.

3.2.1 Структура та алгоритм роботи програми

Програмний засіб було реалізовано на мові Python. Основним компонентом системи є клас EmotionBenchmark, який включає в себе логіку завантаження даних, ініціалізації моделей і збору метрик. Код програми наведено в додатку В.

Алгоритм роботи програми містить такі етапи:

а) ініціалізація та завантаження даних: програмний засіб сканує відповідну директорію з тестовим набором даних FER-2013, що містить 7178 зображень. Зображення проходять попередню обробку (препроцесинг), а потім завантажуються в оперативну пам’ять;

б) тестування моделей: далі іде послідовний інференс для кожної з досліджуваних моделей:

– MobileNetV2 (з навченими в ході дослідження вагами): використовує особливий препроцесинг (зміну розміру до 96 x 96, нормалізація діленням на 255). Для оптимізації використовується прямий виклик графа замість методу .predict(), та попередній «прогін» моделі на «dummy_input»;

– FER Library: використовує каскадну модель, а саме MTCNN для детекції та CNN для класифікації;

– DeepFace: використовує SOTA-бібліотеку, що обертає складні архітектури (в даному випадку – RetinaFace/VGG-Face);

в) збір та візуалізація результатів: для кожної моделі фіксуються передбачені мітки класів і час. За підсумками формуються матриця помилок (Confusion Matrix) та зведена таблиця з метриками Accuracy (Точність) і FPS (Кадри на секунду) (див. рис. 3.6 та рис. 3.7).

```
Запуск тесту: DeepFace...
100%|██████████| 7178/7178 [02:07<00:00, 56.10it/s]

Результати для DeepFace:
   Accuracy: 0.5524
   Avg Time: 17.78 ms
   FPS:      56.24

Classification Report:
              precision    recall  f1-score   support

       angry       0.47      0.42      0.45       958
     disgust       0.57      0.29      0.38       111
        fear       0.38      0.37      0.38      1024
       happy       0.79      0.77      0.78      1774
     neutral       0.45      0.56      0.50      1233
         sad       0.42      0.41      0.42      1247
    surprise       0.76      0.68      0.72       831

    accuracy                           0.55      7178
   macro avg       0.55      0.50      0.52      7178
weighted avg       0.56      0.55      0.55      7178
```

Рисунок 3.6 – Консольний вивід результатів класифікації для моделі DeepFace

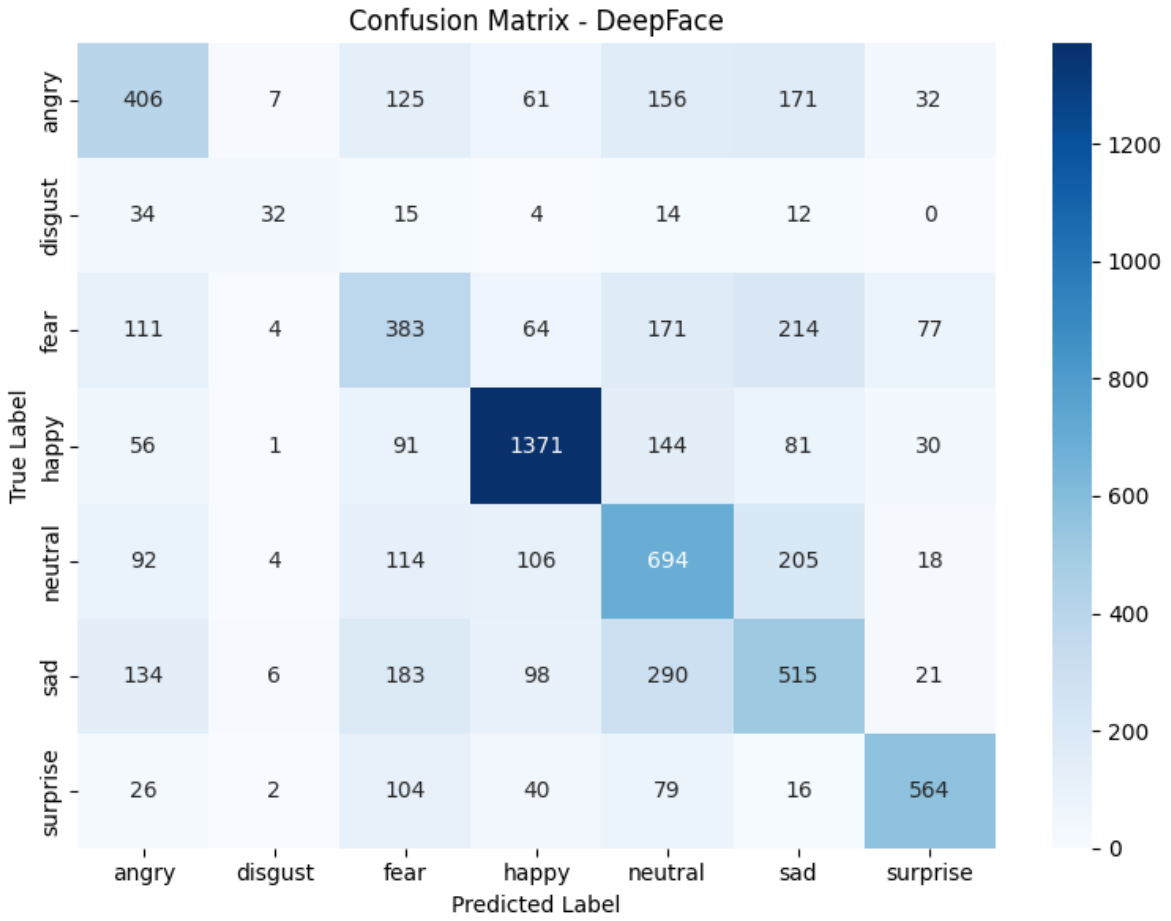


Рисунок 3.7 – Матриця помилок для моделі DeepFace

3.2.2 Результати порівняльного аналізу

Тестування проводилося на повному наборі даних тестової вибірки (а саме 7178 зображень). Результати виводяться у кінці консольного звіту програми (див. рис 3.8).

```
Бенчмарк завершено! Результати збережено у benchmark_results.csv
            Model  Accuracy        FPS  Avg_Time_ms
0  My_MobileNetV2  0.539008   7.702770   129.823421
1     FER_Library  0.409724  92.970137    10.756142
2        DeepFace  0.552382  56.243784    17.779742
```

Рисунок 3.8 – Загальні результати порівняльного аналізу

Проаналізувавши результати, були отримані такі висновки: модель MobileNetV2 з навченими в ході дослідження вагами показала точність 53.9%, що значно перевершує (на 13%) базове рішення – FER Library (Accuracy дорівнює 40.97%), і майже наближається до рівня SOTA-бібліотеки DeepFace (Accuracy дорівнює 55.24%) з відставанням всього на ≈ 1.3%, що є відмінним показником для моделі, навченої з нуля.

Проте показник швидкодії (7.8 FPS порівняно з 56.8 FPS та 92.9 FPS у більш легких моделей) сигналізує про високу обчислювальну складність архітектури при роботі з CPU, що вказує на необхідність подальшої оптимізації.

3.3 Оптимізація моделі з використанням TensorFlow Lite

З огляду на те, що модель MobileNetV2 взагалі то призначена для використання в системах реального часу, показник у 7.8 FPS є недостатнім. Для вирішення цієї проблеми буде застосовано метод квантування та конвертації моделі .h5 у формат .tflite (TensorFlow Lite).

Були реалізовані дві стратегії оптимізації:

– Dynamic Range Quantization (Квантування динамічного діапазону): ваги моделі стискаються до цілочисельного формату (int8), але функції активації обчислюються у форматі з плаваючою комою (float32). Це забезпечує баланс між швидкістю та збереженням точності;
– Full Integer Quantization (Повне цілочисельне квантування): і ваги, і функції активації та вхідні/вихідні дані конвертуються у формат int8. Це дозволяє використовувати цілочисельні інструкції CPU, але вимагає складного калібрування. Такий підхід дає величезний приріст к FPS, але впливає на точність.

3.3.1 Розробка конвертера та калібрування

Для реалізації цих стратегій було розроблено скрипт, код якого представлено в Додатку Г. Для повного квантування було підготовлено репрезентативну вибірку (Representative Dataset). Щоб уникнути падіння точності, був створений генератор даних, який зчитує по випадкових 100 зображень для 7 класів емоцій, та виконана попередня обробка зображень, яка ідентична тій, що використовувалась при навчанні моделі. Це є критично важливим для коректного обчислення масштабних коефіцієнтів (Scale) і нульових точок (Zero Point).

3.3.2 Реалізація універсального модуля інференсу

Оскільки порівнювані версії моделей вимагають різних форматів вхідних даних (float32 для Dynamic Range і int8 для Full Integer), в клас EmotionBenchmark був впроваджений адаптивний механізм інференсу. Метод run_tflite в головній програмі автоматично визначає тип даних вхідного тензора моделі і застосовує відповідну логіку обробки:

– для моделей Float32 (Dynamic Range) використовується стандартна нормалізація зображень (ділення значень пікселів на 255.0);
– для моделей Int8 (Full Integer): Реалізовано алгоритм ручного квантування вхідних даних. Перед подачею в інтерпретатор нормалізоване зображення перетворюється за формулою афінного квантування [23]:

$$q = clamp\left(round\left(\frac{r}{S} + Z\right); -128{,}127\right), \tag{3.1}$$

де q – квантоване значення (int8);

r – реальне значення (float32);

s – масштабний коефіцієнт (Scale), отриманий з метаданих моделі;

Z – нульова точка (Zero Point).

3.3.3 Результати оптимізації

За результатами тестування було отримано порівняльні характеристики продуктивності на тестовому наборі даних (див. рис. 3.9):

```
Бенчмарк завершено! Результати збережено у benchmark_results.csv
            Model  Accuracy         FPS  Avg_Time_ms
0  My_MobileNetV2  0.539008    7.339227   136.254123
1  TFLite_Dynamic  0.530510   64.636079    15.471236
2     TFLite_Int8  0.410003  284.623456     3.513414
3     FER_Library  0.409724   93.356629    10.711612
4        DeepFace  0.552382   55.073500    18.157553
```

Рисунок 3.9 – Результати порівняльного аналізу після оптимізації

Аналіз результатів:

– модель з динамічним квантуванням (Dynamic Quantization) показала найкращий баланс характеристик: при зменшенні розміру файлу в 4 рази і збільшенні швидкості майже в 9 разів, порівняно з оригінальною версією (з 7.3 FPS до 64.6 FPS), точність зменшилась всього на 0.85% (з 53.90% до 53.05%). Більш того, ця модель перевершила модель DeepFace в швидкості (55.07 FPS), зберігши при цьому порівняний рівень точності (різниця ≈ 2%). Результати для моделей MyMobileNetV2 та TFLite Dynamic можна переглянути у вигляді матриць помилок (див. рис. 3.10);

– модель з повним цілочисельним квантуванням (Full Int8) продемонструвала екстремальний приріст в продуктивності – 284 FPS, але жорстке квантування призвело до втрати точності – зменшення до 41%, що відповідає рівню базового алгоритму FER Library. Ця версія

підходить виключно для вбудованих систем з вкрай обмеженими ресурсами (мікроконтролери, IoT), для яких швидкість є критичнішою за точність.

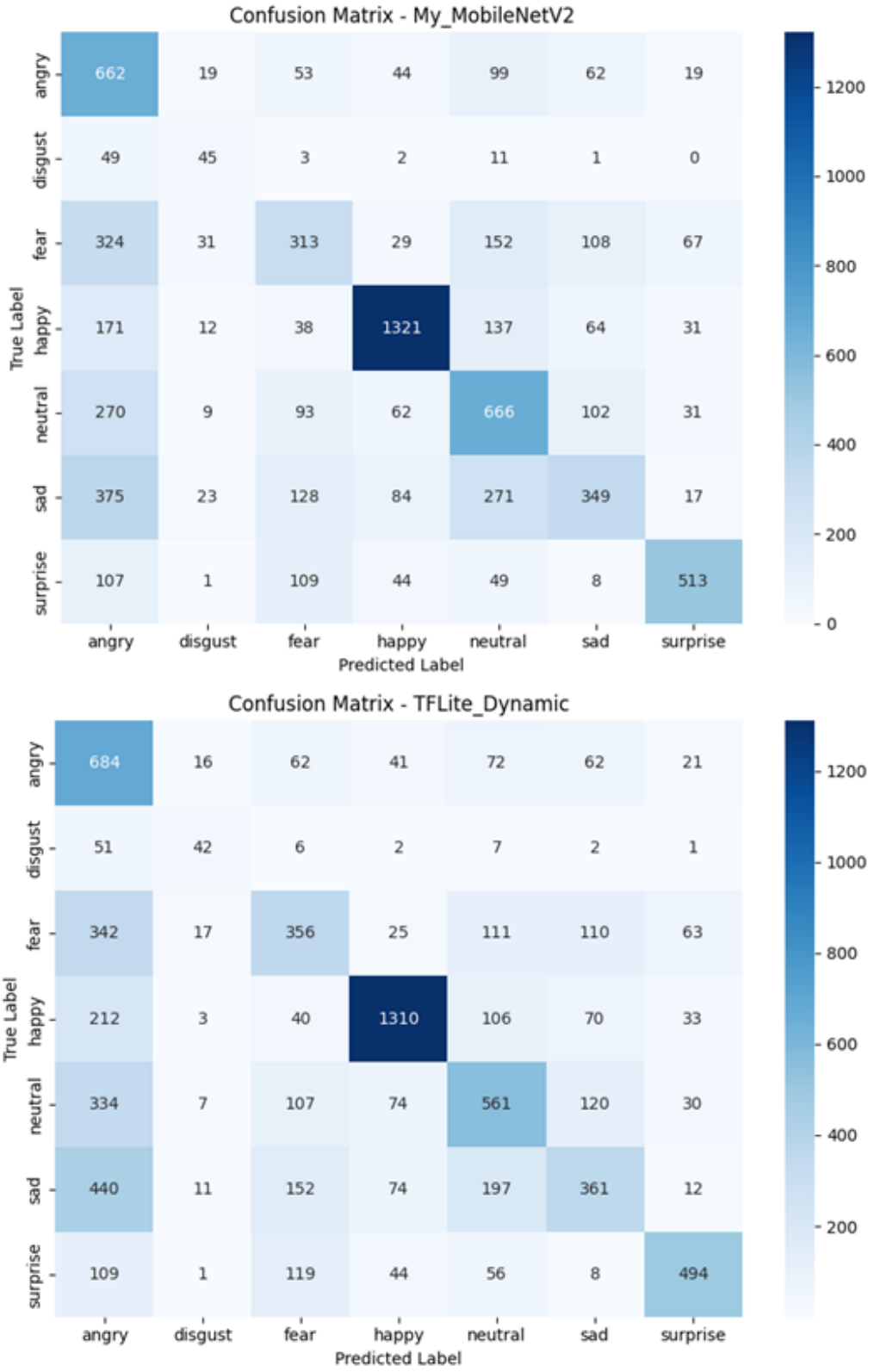


Рисунок 3.10 – Матриці помилок MyMobileNetV2 та TFLite Dynamic для порівняння

ВИСНОВКИ

Мета кваліфікаційної роботи – дослідження та порівняльний аналіз ефективності сучасних технологій комп'ютерного зору для розпізнавання психоемоційного стану людини, була повністю досягнута.

В процесі дослідження були розглянуті основи теорії психофізіологічних характеристик емоцій (модель Пола Екмана, FACS), були проаналізовані існуючі інструменти систем комп'ютерного зору, зокрема бібліотеки FER та DeepFace. Також, вибір архітектури MobileNetV2 в якості основної моделі було обґрунтовано її ефективністю для мобільних та вбудованих систем. Для реалізації практичної частини було обрано мову програмування Python, бібліотеки TensorFlow/Keras та OpenCV.

Загалом був реалізований повний цикл розробки інтелектуальної системи, а саме підготовлений набор даних (датасет FER-2013), була навчена модель за стратегію Fine-Tuning, створена основна програма для бенчмаркінгу та збору метрик (Accuracy, Precision, Recall, F1-score, FPS), розроблені скрипти для навчання моделі, конвертації та перевірки моделі на зображеннях користувача.

Експериментальна частина продемонструвала високу конкурентоспроможність обраного підходу: навчена модель досягла точності 53.90%, що суттєво перевершує базову бібліотеку FER (40.97%) і конкурує з більш ресурсномісткою бібліотекою DeepFace (55.24%).

Окрему увагу було приділено оптимізації швидкодії для роботи з CPU та в режимі реального часу. Було досліджено та застосовано методи квантування (Dynamic Range та Full Integer Quantization) з конвертацією моделі у формат TensorFlow Lite. Це дозволило зменшити розмір моделі у 4 рази та підвищити швидкість обробки на CPU з 7 FPS до 64 FPS (для динамічного квантування) та

навіть до 284 FPS (для повного квантування), що робить систему придатною для використання на пристроях з обмеженими ресурсами.

В результаті дослідження було доведено, що сучасні методи оптимізації нейромереж дозволяють створювати високопродуктивні системи розпізнавання емоцій, які можуть бути інтегровані у різні сфери людського життя: від освітніх платформ та медичних додатків до систем моніторингу безпеки та покращення клієнтського сервісу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ekman P. An argument for basic emotions. *Cognition and Emotion*. 1992. Vol. 6, no. 3-4. P. 169–200. URL : https://doi.org/10.1080/02699939208411068 (date of access : 15.07.2025).

2. Darwin C. The Expression of the Emotions in Man and Animals. London : John Murray, 1872. URL : https://www.gutenberg.org/ebooks/1227 (date of access : 15.07.2025).

3. Ekman P., Friesen W. V. Facial Action Coding System : A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, 1978.

4. Russell J. A. A circumplex model of affect. *Journal of Personality and Social Psychology*. 1980. Vol. 39, no. 6. P. 1161–1178. URL : https://doi.org/10.1037/h0077714 (date of access : 15.07.2025).

5. Martinez A. M. Computational Models of Face Perception. *Current Directions in Psychological Science*. 2017. Vol. 26, no. 3. P. 263–269. URL : https://doi.org/10.1177/0963721417702108 (date of access : 15.07.2025).

6. Tian Y. I., Kanade T., Cohn J. F. Facial expression analysis. Handbook of face recognition. 2005. P. 247–275.

7. Zeng Z. et al. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE transactions on pattern analysis and machine intelligence*. 2009. Vol. 31, no. 1. P. 39–58. URL : https://doi.org/10.1109/TPAMI.2008.52 (date of access : 15.07.2025).

8. Shan C., Gong S., McOwan P. W. Facial expression recognition based on Local Binary Patterns : A comprehensive study. *Image and Vision Computing*. 2009. Vol. 27, no. 6. P. 803–816. URL : https://doi.org/10.1016/j.imavis.2008.08.005 (date of access : 15.07.2025).

9. Dalal N., Triggs B. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern*

Recognition (CVPR). 2005. Vol. 1. P. 886–893. URL : https://doi.org/10.1109/CVPR.2005.177 (date of access : 15.07.2025).

10. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012. Vol. 25. P. 1097–1105.

11. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*. 2014. URL : https://arxiv.org/abs/1409.1556 (date of access : 15.07.2025).

12. He K. et al. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. P. 770–778. URL : https://doi.org/10.1109/CVPR.2016.90 (date of access : 15.07.2025).

13. Howard A. G. et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*. 2017. URL : https://arxiv.org/abs/1704.04861 (date of access : 15.07.2025).

14. Ng H. W. et al. Deep learning for emotion recognition on small datasets using transfer learning. *Proceedings of the 17th ACM international conference on multimodal interaction*. 2015. P. 443–449. URL : https://doi.org/10.1145/2818346.2830593 (date of access : 15.07.2025).

15. Lucey P. et al. The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition – Workshops*. 2010. P. 94–101. URL : https://doi.org/10.1109/CVPRW.2010.5543262 (date of access : 15.07.2025).

16. Goodfellow I. J. et al. Challenges in representation learning: A report on three machine learning contests. *International Conference on Neural Information Processing Systems*. 2013.

17. Mollahosseini A., Hasani B., Mahoor M. H. AffectNet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*. 2017. Vol. 10, no. 1. P. 18–31. URL : https://doi.org/10.1109/TAFFC.2017.2740923 (date of access : 15.07.2025).

18. Serengil S. I., Ozpinar A. LightFace: A Hybrid Deep Face Recognition Framework. *Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2020. P. 1–5. URL : https://doi.org/10.1109/ASYU50717.2020.9259802 (date of access : 15.07.2025).

19. Gonzalez R. C., Woods R. E. Digital Image Processing. 4th ed. Pearson, 2018. 1168 p.

20. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*. 2015. P. 448–456. URL : http://proceedings.mlr.press/v37/ioffe15.html (date of access : 10.10.2025).

21. Grandini M., Bagli E., Visani G. Metrics for Multi-Class Classification: an Overview. *arXiv preprint arXiv :2008.05756*. 2020. URL : https://arxiv.org/abs/2008.05756 (date of access : 10.10.2025).

22. Sze V. et al. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*. 2017. Vol. 105, no. 12. P. 2295–2329. URL : https://doi.org/10.1109/JPROC.2017.2761740 (date of access : 10.10.2025).

23. Jacob B. et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018. P. 3060–3071. URL : https://doi.org/10.1109/CVPR.2018.00323 (date of access : 10.10.2025).

ДОДАТОК А

Код підпрограми для навчання моделі

```
import os
import matplotlib.pyplot as plt
import tensorflow as tf

TRAIN_DIR = ‘datasets/fer2013/train’
VAL_DIR = ‘datasets/fer2013/test’
SAVE_MODEL_NAME = ‘mobilenet_fer2013_finetuned2.h5’

IMG_SIZE = 96
BATCH_SIZE = 32
EPOCHS_HEAD = 15
EPOCHS_FINE = 25
TOTAL_EPOCHS = EPOCHS_HEAD + EPOCHS_FINE



def train_model():
    print(f"TF Version: {tf.__version__}")

    train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
        rescale=1. / 255,
        rotation_range=10,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True,
```

```
    shear_range=0.05,
    zoom_range=0.1,
    brightness_range=[0.8, 1.2],
    fill_mode='nearest'
)

val_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1. / 255)

print("\nЗавантаження данных...")
train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    color_mode='rgb',
    shuffle=True
)

validation_generator = val_datagen.flow_from_directory(
    VAL_DIR,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    color_mode='rgb',
    shuffle=False
)

print(f"\nЗнайдено класів: {train_generator.num_classes}")
print(f"Навчальних зображень: {train_generator.samples}")
print(f"Валідаційних зображень: {validation_generator.samples}")
```

```
print(f"Класи: {train_generator.class_indices}")

print("\nЗавантаження MobileNetV2...")
base_model = tf.keras.applications.MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(IMG_SIZE, IMG_SIZE, 3)
)

# --- Заморожування (Training Head) ---
base_model.trainable = False

x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(256, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
predictions = tf.keras.layers.Dense(7, activation='softmax')(x)

model = tf.keras.models.Model(inputs=base_model.input, outputs=predictions)

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

reduce_lr_head = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=3,
```

```
    min_lr=1e-7,
    verbose=1
)

checkpoint_head = tf.keras.callbacks.ModelCheckpoint(
    'mobilenet_fer2013_head.h5',
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)

print("\nЕТАП 1: Head...")
print(f"   Learning Rate: 0.0005 (з автоматичним зниженням)")
history_head = model.fit(
    train_generator,
    epochs=EPOCHS_HEAD,
    validation_data=validation_generator,
    callbacks=[reduce_lr_head, checkpoint_head]
)

print(f"\nРезультати Етапу 1:")
print(f"   Найкраща точність валідації:
{max(history_head.history['val_accuracy']):.4f}")
print(f"   Фінальна точність валідації: {history_head.history['val_accuracy'][-
1]:.4f}")

# --- Fine-Tuning ---
print("\nЕТАП 2: Fine-Tuning...")

base_model.trainable = True
```

```
    # Розморожування останніх 50 шарів
    fine_tune_at = 100

    for layer in base_model.layers[:fine_tune_at]:
        layer.trainable = False

    print(f"   Кількість шарів базової моделі: {len(base_model.layers)}")
    print(f"   Буде навчатись шарів: {len(base_model.layers) – fine_tune_at}
(починаючи з {fine_tune_at}-го)")

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    reduce_lr_fine = tf.keras.callbacks.ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.5,
        patience=4,
        min_lr=1e-7,
        verbose=1
    )

    checkpoint = tf.keras.callbacks.ModelCheckpoint(
        SAVE_MODEL_NAME,
        monitor='val_accuracy',
        save_best_only=True,
        verbose=1
```

```
    )

    early_stop = tf.keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=8,
        restore_best_weights=True,
        verbose=1
    )

    print(f"\n   Learning Rate для fine-tuning: 5e-5 (з автоматичним зниженням)")

    history_fine = model.fit(
        train_generator,
        initial_epoch=EPOCHS_HEAD,
        epochs=TOTAL_EPOCHS,
        validation_data=validation_generator,
        callbacks=[checkpoint, early_stop, reduce_lr_fine]
    )

    print(f"\nРезультати Етапу 2 (Fine-Tuning):")
    print(f"   Найкраща точність валідації:
{max(history_fine.history['val_accuracy']):.4f}")
    print(f"   Фінальна точність валідації: {history_fine.history['val_accuracy'][-
1]:.4f}")

    plot_history(history_head, history_fine)


def plot_history(history1, history2):
    acc = history1.history['accuracy'] + history2.history['accuracy']
```

```
    val_acc = history1.history['val_accuracy'] + history2.history['val_accuracy']
    loss = history1.history['loss'] + history2.history['loss']
    val_loss = history1.history['val_loss'] + history2.history['val_loss']

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.plot(acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.axvline(x=EPOCHS_HEAD, color='green', linestyle='--', label='Start Fine Tuning')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(1, 2, 2)
    plt.plot(loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.axvline(x=EPOCHS_HEAD, color='green', linestyle='--', label='Start Fine Tuning')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')

    plt.savefig('training_history_finetuned.png')
    plt.show()


if __name__ == "__main__":
    train_model()
```

ДОДАТОК Б

Код підпрограми для тестування навченої моделі на користувацьких зображеннях

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

MODEL_PATH = ‘mobilenet_fer2013_finetuned2.h5’
IMAGE_PATH = ‘test_image.jpg’
IMG_SIZE = 96

EMOTIONS = {
    0: ‘Angry (Злість)’,
    1: ‘Disgust (Відраза)’,
    2: ‘Fear (Страх)’,
    3: ‘Happy (Радість)’,
    4: ‘Sad (Сум)’,
    5: ‘Surprise (Здивування)’,
    6: ‘Neutral (Нейтрально)’
}



def predict_emotion(image_path, model_path):
    if not os.path.exists(image_path):
```

```
    print(f"Помилка: файл зображення не знайдено: {image_path}")
    return

try:
    print("Завантаження моделі...")
    model = tf.keras.models.load_model(model_path)
    print("Модель завантажена успішно")

    img = cv2.imread(image_path)
    if img is None:
        print("Не вдалося прочитати зображення через OpenCV")
        return

    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_resized = cv2.resize(img_rgb, (IMG_SIZE, IMG_SIZE))

    plt.imshow(img_resized)
    plt.axis('off')
    plt.title("Input Image")
    plt.show()

    img_array = np.expand_dims(img_resized, axis=0)
    img_array = img_array.astype('float32')
    img_processed = tf.keras.applications.mobilenet.preprocess_input(img_array)

    predictions = model.predict(img_processed)

    predicted_class = np.argmax(predictions)
    confidence = np.max(predictions)
```

```
        print("\n" + "=" * 40)
        print(f"РЕЗУЛЬТАТ: {EMOTIONS[predicted_class]}")
        print(f"Впевненість: {confidence:.2%}")
        print("=" * 40)

        print("\nДетальний розподіл ймовірностей:")
        print("-" * 40)



        for i, prob in enumerate(predictions[0]):
            emotion_name = EMOTIONS[i]
            print(f"{emotion_name:<25} : {prob:.4f} ({prob:.2%})")

        print("-" * 40 + "\n")

    except Exception as e:
        print(f"Критична помилка: {e}")



if __name__ == "__main__":
    predict_emotion(IMAGE_PATH, MODEL_PATH)
```

ДОДАТОК В

Код основного програмного модуля для бенчмаркінгу

```
import os
import cv2
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import tensorflow as tf

from fer.fer import FER
from deepface import DeepFace

DATASET_PATH = "datasets/fer2013/test"
RESULTS_FILE = "benchmark_results.csv"
MY_MODEL_PATH = "mobilenet_fer2013_finetuned2.h5"

EMOTIONS_MAP = {
    'angry': 0,
    'disgust': 1,
    'fear': 2,
```

```
    'happy': 3,
    'neutral': 4,
    'sad': 5,
    'surprise': 6
}


class EmotionBenchmark:
    def __init__(self, dataset_path):
        self.dataset_path = dataset_path
        self.images = []
        self.labels = []
        self.filenames = []

        print(f"Ініціалізація бенчмарку...")
        self.load_data()

    def load_data(self):
        print(f"Завантаження датасету з {self.dataset_path}...")

        if not os.path.exists(self.dataset_path):
            print(f"ПОМИЛКА: Папка {self.dataset_path} не існує!")
            return

        for emotion_name, label_idx in EMOTIONS_MAP.items():
            folder = os.path.join(self.dataset_path, emotion_name)
            if not os.path.exists(folder):
                print(f"Увага: Папка {folder} не знайдена!")
                continue
```

```
        files = os.listdir(folder)

        # Обмеження вибірки для швидкого тесту
        # files = files[:100]

        for file in tqdm(files, desc=f"Loading {emotion_name}"):
            img_path = os.path.join(folder, file)
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

            if img is not None:
                img = cv2.resize(img, (48, 48))
                self.images.append(img)
                self.labels.append(label_idx)
                self.filenames.append(file)

    print(f"Завантажено {len(self.images)} зображень.")

def run_fer_library(self):
    print("\nЗапуск тесту: FER Library (MTCNN)...")
    try:
        detector = FER(mtcnn=True)
    except Exception as e:
        print(f"Помилка ініціалізації FER: {e}")
        return None

    y_true = []
    y_pred = []
    inference_times = []

    for i, img_gray in enumerate(tqdm(self.images)):
```

```
        img_rgb = cv2.cvtColor(img_gray, cv2.COLOR_GRAY2RGB)

        start_time = time.time()
        try:
            result = detector.detect_emotions(img_rgb)
        except:
            result = []
        end_time = time.time()

        inference_times.append((end_time – start_time) * 1000)

        y_true.append(self.labels[i])
        if result:
            top_emotion = max(result[0][‘emotions’], key=result[0][‘emotions’].get)
            y_pred.append(EMOTIONS_MAP.get(top_emotion, 6))
        else:
            y_pred.append(6)

    return self._generate_report("FER_Library", y_true, y_pred, inference_times)

def run_deepface(self):
    print("\nЗапуск тесту: DeepFace...")

    y_true = []
    y_pred = []
    inference_times = []

    for i, img_gray in enumerate(tqdm(self.images)):
        img_rgb = cv2.cvtColor(img_gray, cv2.COLOR_GRAY2RGB)
```

```
            start_time = time.time()
            try:
                result = DeepFace.analyze(img_rgb, actions=['emotion'],
enforce_detection=False)

                if isinstance(result, list): result = result[0]
                top_emotion = result['dominant_emotion']
            except Exception as e:
                print(f"DeepFace Error: {e}")
                top_emotion = 'neutral'

            end_time = time.time()

            inference_times.append((end_time – start_time) * 1000)
            y_true.append(self.labels[i])
            y_pred.append(EMOTIONS_MAP.get(top_emotion, 6))

        return self._generate_report("DeepFace", y_true, y_pred, inference_times)

    def run_mobilenet(self):
        print(f"\nЗапуск тесту: My MobileNetV2 ({MY_MODEL_PATH})...")

        if not os.path.exists(MY_MODEL_PATH):
            print(f"Помилка: файл моделі {MY_MODEL_PATH} не знайдено!")
            return None

        try:
            model = tf.keras.models.load_model(MY_MODEL_PATH)
            print("   Модель успішно завантажена.")
        except Exception as e:
```

```
            print(f"Помилка завантаження моделі: {e}")
            return None

        y_true = []
        y_pred = []
        inference_times = []

        # Створюємо тестовий прогін, щоб TensorFlow ініціалізував граф, це прибере затримку на першому кадрі
        dummy_input = tf.zeros((1, 96, 96, 3))
        _ = model(dummy_input, training=False)
        print("Модель готова до роботи.")

        for i, img_gray in enumerate(tqdm(self.images)):
            img_rgb = cv2.cvtColor(img_gray, cv2.COLOR_GRAY2RGB)
            img_resized = cv2.resize(img_rgb, (96, 96))

            img_array = np.expand_dims(img_resized, axis=0)
            img_array = img_array.astype('float32')
            img_processed = img_array / 255.0

            start_time = time.time()

            predictions = model(img_processed, training=False)

            predictions = predictions.numpy()

            end_time = time.time()
            inference_times.append((end_time – start_time) * 1000)
```

```
            predicted_class = np.argmax(predictions)

            y_true.append(self.labels[i])
            y_pred.append(predicted_class)

        return self._generate_report("My_MobileNetV2", y_true, y_pred,
inference_times)

    def run_tflite(self, tflite_path):
        print(f"\nЗапуск тесту: TFLite ({tflite_path})...")

        if not os.path.exists(tflite_path):
            print(f"Файл {tflite_path} не знайдено! Спочатку запустіть конвертацію.")
            return None

        interpreter = tf.lite.Interpreter(model_path=tflite_path)
        interpreter.allocate_tensors()

        input_details = interpreter.get_input_details()[0]
        output_details = interpreter.get_output_details()[0]

        input_dtype = input_details['dtype']

        input_scale, input_zero_point = input_details['quantization']
        output_scale, output_zero_point = output_details['quantization']

        y_true = []
        y_pred = []
        inference_times = []
```

```
print(f"   Input Type: {input_dtype}")

for i, img_gray in enumerate(tqdm(self.images)):
    img_rgb = cv2.cvtColor(img_gray, cv2.COLOR_GRAY2RGB)
    img_resized = cv2.resize(img_rgb, (96, 96))

    img_float = img_resized.astype('float32') / 255.0
    img_float = np.expand_dims(img_float, axis=0)

    if input_dtype == np.float32:
        img_input = img_float

    elif input_dtype == np.int8:
        img_int8 = (img_float / input_scale) + input_zero_point
        img_input = np.clip(img_int8, -128, 127).astype(np.int8)
    else:
        print("Невідомий тип даних моделі!")
        return None

    start_time = time.time()

    interpreter.set_tensor(input_details['index'], img_input)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details['index'])

    end_time = time.time()
    inference_times.append((end_time – start_time) * 1000)

    if output_scale > 0:
        prediction_float = (output_data.astype('float32') – output_zero_point) *
```

```
output_scale
            else:
                prediction_float = output_data

            predicted_class = np.argmax(prediction_float)
            y_true.append(self.labels[i])
            y_pred.append(predicted_class)

        report_name = "TFLite_Dynamic" if input_dtype == np.float32 else
"TFLite_Int8"

        return self._generate_report(report_name, y_true, y_pred, inference_times)

    def _generate_report(self, model_name, y_true, y_pred, times):
        acc = accuracy_score(y_true, y_pred)
        avg_time = np.mean(times)
        fps = 1000 / avg_time if avg_time > 0 else 0

        print(f"\nРезультати для {model_name}:")
        print(f"   Accuracy: {acc:.4f}")
        print(f"   Avg Time: {avg_time:.2f} ms")
        print(f"   FPS:      {fps:.2f}")

        report = classification_report(y_true, y_pred,
target_names=EMOTIONS_MAP.keys(), zero_division=0)
        print("\nClassification Report:")
        print(report)

        with open(f"report_{model_name}.txt", "w") as f:
            f.write(report)
```

```
        # ------------------------------------------------

        cm = confusion_matrix(y_true, y_pred)
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                    xticklabels=EMOTIONS_MAP.keys(),
                    yticklabels=EMOTIONS_MAP.keys())
        plt.title(f'Confusion Matrix – {model_name}')
        plt.ylabel('True Label')
        plt.xlabel('Predicted Label')
        plt.tight_layout()
        plt.savefig(f"confusion_matrix_{model_name}.png")
        plt.close()

        return {
            "Model": model_name,
            "Accuracy": acc,
            "FPS": fps,
            "Avg_Time_ms": avg_time
        }


if __name__ == "__main__":
    benchmark = EmotionBenchmark(DATASET_PATH)
    results = []

    # --- MobileNetV2 ---
    res_mobile = benchmark.run_mobilenet()
    if res_mobile: results.append(res_mobile)
```

```
# --- TFLite Dynamic ---
res_dynamic = benchmark.run_tflite('mobilenet_quantized_dynamic.tflite')
if res_dynamic: results.append(res_dynamic)

# --- TFLite Int8 ---
res_int8 = benchmark.run_tflite('mobilenet_quantized_int8.tflite')
if res_int8: results.append(res_int8)

# --- FER Library ---
res_fer = benchmark.run_fer_library()
if res_fer: results.append(res_fer)

# --- DeepFace ---
res_deepface = benchmark.run_deepface()
if res_deepface: results.append(res_deepface)

if results:
    df = pd.DataFrame(results)
    df.to_csv(RESULTS_FILE, index=False)
    print(f"\nБенчмарк завершено! Результати збережено у {RESULTS_FILE}")
    print(df)
else:
    print("\nЖодних результатів не отримано.")
```

ДОДАТОК Г

Код підпрограми для конвертації моделі у формат tflite

```
import tensorflow as tf
import numpy as np
import os
import cv2

MODEL_PATH = ‘mobilenet_fer2013_finetuned2.h5’
TFLITE_PATH = ‘mobilenet_quantized_int8.tflite’
DATASET_DIR = ‘datasets/fer2013/test’

EMOTIONS = [‘angry’, ‘disgust’, ‘fear’, ‘happy’, ‘neutral’, ‘sad’, ‘surprise’]


def representative_data_gen():
    print("Генерація калібрувальних даних...")

    for emotion in EMOTIONS:
        folder = os.path.join(DATASET_DIR, emotion)
        if not os.path.exists(folder): continue

        files = os.listdir(folder)[:20]

        for file in files:
            img_path = os.path.join(folder, file)
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
```

```
        if img is None: continue

        img_rgb = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
        img_resized = cv2.resize(img_rgb, (96, 96))
        img_array = np.expand_dims(img_resized, axis=0)
        img_array = img_array.astype('float32')
        img_processed = img_array / 255.0

        yield [img_processed]


print("Загрузка модели...")
model = tf.keras.models.load_model(MODEL_PATH)

converter = tf.lite.TFLiteConverter.from_keras_model(model)

converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]

converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8

print("Початок конвертації...")
tflite_model = converter.convert()

with open(TFLITE_PATH, 'wb') as f:
    f.write(tflite_model)
```

```
print(f"Конвертація успішна. Модель: {TFLITE_PATH}")
print(f"Розмір: {len(tflite_model) / 1024:.2f} KB")
```