Одеський національний політехнічний університет

Інститут комп'ютерних систем

Комп'ютерні системи

Стеценко Ігор Володимирович

# КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

## ДОСЛІДЖЕННЯ МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ СИСТЕМИ РОЗПІЗНАВАННЯ ЛЮДСЬКИХ ЕМОЦІЙ

Спеціальність 123 – Комп'ютерна інженерія

Спеціалізація — Спеціалізовані комп'ютерні системи

Керівник: Яковлев Дмитро Павлович,

кандидат технічних наук, професор

Одеса – 2020

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ANFIS – adaptive neuro-fuzzy inference system.

HMM – hidden Markov model.

MANFIS – modified adaptive neuro-fuzzy inference system.

ЕОМ – електронно-обчислювальна машина.

РЕСЛ – розпізнавання емоційного стану людини.

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ІС – інформаційна система

GUI (graphical user interface) – графічний користувальницький інтерфейс

DTW (dynamic time wraping) – алгоритм динамічної трансформації часу

LBP (local binary patterns) – локальні бінарні шаблони

AAM (active appearance model) – активні моделі зовнішнього вигляду

FACS (Facial Action Coding System) – система кодування рухів обличчя

# ВСТУП

В даний час бурхливо розвивається галузь робототехніки що займається створенням мобільних роботів. Дедалі популярнішими стають сервісні мобільні роботи. До даного типу можна віднести роботів-екскурсоводів, роботів-пилососів, рухливі інформаційні термінали, беспілотні автомобілі, тощо. Для виконання своїх функцій ці роботи повинні вміти розпізнавати та класифікувати об’єкти навколишнього середовища в автоматичному режимі.

У наш час інформаційні технології досягли значного розвитку й продовжують розвиватися. Разом із тим, підвищується актуальність задач розпізнавання та класифікації образів.

Автоматичне розпізнавання та аналіз виразів обличчя є активною темою в наукових дослідженнях вже більше двох десятиліть [1]. Нещодавні психологічні дослідження показали, що міміка є найбільш точним способом виявлення емоцій у людей. Словесна частина повідомлення дорівнює лише 7% ефекту повідомлення в цілому, вокальна частина - 38%, а вираз обличчя - 55% від ефекту повідомлення спікера [2].

Тому автоматичне розпізнавання обличчя та його емоціонального виразу в режимі реального часу може бути корисним для багатьох галузей, наприклад віртуальної реальності, відеоконференцій, дослідження задоволеності клієнтів тощо. Інформаційна система, що сприймає людські емоції, забезпечує ліпший інтерфейс взаємодії робота з людиною, адаптується під її настрій і стан. Емоції можуть стати додатковим невербальним фільтром під час пошуку інформації, бути причиною зміни ігрового, або тренувального процесу, чи братись до уваги під час оцінки істинності вербальної інформації з боку роботизірованої системи.

Саме розпізнавання людських емоцій може стати основною частиною проекту, де на додаток до біометричних параметрів оточення робота, додається інформація про міміку, що грає важливу роль при вирішенні такого завдання, як розпізнати настрій людини. Хоча для людини досить просто виявляти та розуміти емоційний стан інших людей, проте точність розпізнавання емоцій машиною залишається проблемою.

На сьогоднішній день розроблено низку комерційних програмних систем, які розпізнають емоції за зображенням обличчя. Проте, вони мають певні обмеження, такі як незадовільну похибку розпізнавання, або ж дуже високу вартість, тому актуальність розпізнавання емоцій людини на обличчі людини залишається на досить високому рівні.

**Об’єкт дослідження** даної роботи є процесс розпізнавання емоцій людини за фотографічним зображенням його обличчя з різними мімічними проявами емоцій.

**Предметом дослідження** виступають методи та алгоритми аналізу мімічних проявів емоцій на фотографічному зображенні обличчя людини.

**Метою роботи** є дослідження та розробка методики розпізнавання емоцій на обличчі людини у відеопотоці. Для вирішення поставленої мети було визначено такі **завдання**:

– аналіз стану проблеми та аналітичний огляд наукових досліджень в області розпізнавання емоцій на обличчі людини;

– дослідження алгоритмів розпізнавання емоцій на обличчі людини;

– визначення методики розпізнавання емоційного стану людини за зображенням обличчя;

– використання запропонованої методики для розробки моделі системи розпізнавання емоцій;

– тестування роботи моделі та аналіз результатів дослідження.

**Наукова новизна** дослідження полягає у подальшому розвитку моделі та методики розпізнавання емоцій на обличчі людини, розширення існучих алгоримів з використанням різних типів ознак.

**Практична цінність** результатів дослідження полягає у можливості їх застосування у системах розпізнавання емоції з підвищенням точності та спрощенням обчислень.

**Публікації**. За результатами дослідження підготовлена наукова стаття для публікації у фаховому виданні.

# 1 АНАЛІТИЧНИЙ ОГЛЯД

## 1.1 Сучасний стан проблеми розпізнавання емоції людини.

Міжособистісне спілкування людей передбачає не тільки розмовну мову, але й невербальні сигнали – такі, як жести, вираз обличчя і тон голосу, які використовуються для висловлення почуттів і надання зворотного зв'язку. Сучасні інтерфейси комп’ютер–людина не використовують інформацію з невербальних каналів, у результаті чого часто взаємодія є менш ефективною.

Здатність комп’ютерів розпізнавати такого роду інформацію могла б надавати користувачам інформацію, яка б найбільше відображала їхні потреби та особливості.

Більшість методів комп'ютерного розпізнавання емоцій засноване на навчанні з учителем. В якості вихідних даних в них використовуються або окремі зображення особи, або послідовність кадрів з відеопотоку, в зв'язку з чим методи можна умовно розділити на статичні і динамічні.

Динамічні методи, як правило, використовують інформацію про рух лицьових м'язів в часі. Так, наприклад, один з найперших методів автоматичного розпізнавання емоцій використав оптичні потоки для визначення напрямку і швидкості руху окремих частин особи, які потім зіставлялися із зразками для кожної з розглянутих емоцій [3].

Аналогічний підхід зустрічається і в більш пізніх роботах, однак замість оптичного потоку частіше використовується модель з'єднаних вібрацій, що дозволяє більш точно визначити деформації частин обличчя [4]. Слід зазначити, що існують також напрацювання з автоматичного виділення патернів. Деякі автори використовують приховані моделі для автоматичної сегментації аудіо та видеопослідовностей з подальшою розміткою отриманих сегментів [5].

Одним з головних переваг динамічних методів є те, що вони дозволяють відстежувати короткочасні зміни на людському обличчі - так звані мікроемоції [6]. Однак у багатьох практичних завданнях (наприклад, розпізнавання по одному зображенню) інформація про динаміку відсутня. Тому також часто використовуються статичні методи, засновані виключно на візуальній оцінці.

Хоча існує безліч способів моделювання емоцій (наприклад, криві Безьє [7]), найбільшого поширення при аналізі емоцій отримали активні моделі [8]. Даний клас моделей дозволяє ефективно обчислити положення ключових точок, таких як центр зіниці ока, куточки губ, контур носа і т. д., а потім вже на їх основі побудувати розпізнавання. Найбільш поширеними методами розпізнавання у активних моделях є: евклідова відстань; моделі гауссових сумішей; метод опорних векторів.

Системи розпізнавання емоцій переважно грунтуються на:

– розпізнаванні емоцій за голосом;
– розпізнаванні емоцій за мімікою;
– розпізнаванні емоцій за голосом та мімікою.

Методи, що ґрунтуються на розпізнаванні мови, використовують глобальні ознаки акустичних сигналів для розпізнавання емоції. Наприклад, середнє значення, стандартне відхилення, максимальний та мінімальний тонові контури [9].

На відміну від підходів до аналізу акустичних даних, ознаки, які переважно використовуються у методах розпізнавання емоцій за мімікою, це як правило, просторове положення або переміщення певних точок чи регіонів на обличчі.

Хоча у існуючих методах і моделях досягнуто достатньо високої точності розпізнавання, всі вони мають певні недоліки:

– закритість системи – один із найпоширеніших недоліків, які не дають змоги дослідити систему. Про неї доводиться судити лише за результатами публікацій;

– вимога фізичного контакту із системою (у разі застосування різного роду зчитувачів);

– низька адаптивність – наприклад, під час розпізнавання емоцій людей різних національностей та культур.

Психологи вважають, що будь-яку емоцію людини можна розкласти на комбінацію шести базових емоцій: гнів, відраза, страх, радість, смуток, здивування [10].

На основі проведених дослідженнь можна виділити такі елементи обличчя, розміщення яких важливе для визначення присутності чи відсутності базової емоції (табл. 1.1), [11].

Таблиця 1.1 – Характеристики відомих методів роспізнавання емоцій

| Автори | Основа | Точність | Особливості |
|---|---|---|---|
| Мейс К. | Основні напрямки м’язів обличчя. Правило К найближчих сусідів | 80% | Чотири емоції – радість, гнів огида та подив, |
| Якоб Я. | Відображення рухів, пов’язаних з краєм рота, очей і брів | 88% | Шість основних емоцій |
| Тіан Т. та ін. | Геометричні моделі для виявлення постійних та змінних рис обличчя | 94% | Розпізнає так звані одиниці дії (Action Units) |
| Есса І. та ін. | Параметрична модель незалежних груп м’язів обличчя, просторові-часові шаблони | 95% | Відсутність емоції “сум” |
| Тіан Т. та ін. | Векторний простір мімічних проявів, шаблони контурів основних зон обличчя, B-сплайн апроксимація | 91% | Вісім базових емоцій: горе, радість, страх, надія, гнів, задоволення, інтерес, зневага |

## 1.2 Огляд систем розпізнавання емоцій

На сьогодняшній день існує велика кількість методів виявлення обличь. Виділяють чотири категорії методів виявлення облич (рис.1.1.)

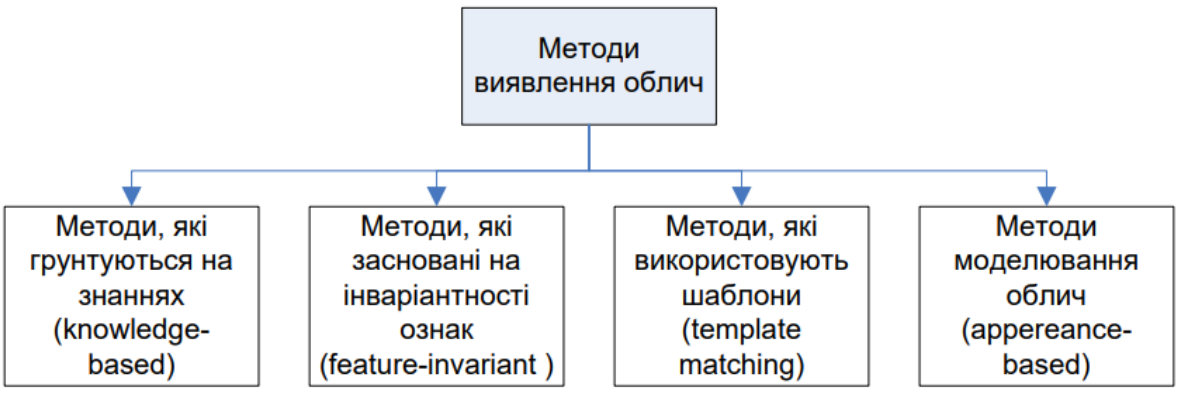


Рисунок 1.1 – Класифікація методів виявлення облич

Перші три категорії методів роблять спробу визначити і використати принципи, якими керується мозок при вирішенні задачі виявлення обличчя. Методи перших трьох

категорій, як правило, використовуються для локалізації обличчя на зображеннях високої якості. Ці методи стійкі до різних умов освітлення, але мають високу обчислювальну вартість [12]. На відміну від перших трьох категорій методи, що ґрунтуються на моделюванні обличчя підходять до проблеми виявлення облич з іншого боку. Ці методи не намагаються у явному вигляді формалізувати процеси, що відбуваються в людському мозку, а намагаються виявити закономірності і методи виявлення облич.

Методи, які засновані на інваріантності ознак (feature-invariant) властивості зображення обличчя неявно, застосовуючи методи математичної статистики і машинного навчання. Методи цієї категорії спираються на інструментарій розпізнавання образів, розглядаючи задачу виявлення обличчя, як окремий випадок задачі розпізнавання. Вони беруть за основу статистичні методи навчання, щоб побудувати класифікатор «обличчя»/«не обличчя» з навчальних прикладів. Ці методи, як правило, використовуються для виявлення облич на зображеннях не дуже високої роздільної здатності [13].

Зі швидким збільшенням обчислювальних ресурсів та пам’яті комп’ютерів методи моделювання обличчя стали домінувати при створенні детекторів облич. Загальна практика полягає у зборі великого набору прикладів «облич» і «не облич», та застосування певних алгоритмів машинного навчання, щоб навчити модель обличчя здійснювати бінарну класифікацію [14].

Існує велика кількість методів фільтрації зображень у складних умовах освітлення для систем відеоспостереження. Усі ці підходи для покращення зображень поділяється на дві великі категорії: методи оброблення у просторовій області і методи оброблення у частотній області. Категорія, методів оброблення у просторовій області, поєднує підходи, засновані на прямому маніпулюванні пікселями зображення.

Методи оброблення в частотній області ґрунтуються на модифікації сигналу, сформованого шляхом застосування до зображення перетворення Фур'є. Також не є другорядними і технології, що базуються на різних комбінаціях методів з цих двох категорій [15]. При роботі в темний час доби ефективність системи відеоспостереження знижується за рахунок того, що різко зростає шум сигналу, який веде до того, що на зображенні з'являється зернистість. А це призводить до того, що відбувається збільшення

бітрейту з причини поганого освітлення, і, отже, виходить погане ущільнення відзнятого матеріалу [16]. Нейронні мережі можуть бути з успіхом застосовані для виявлення облич. При цьому якість розпізнавання знижується при збільшенні кількості класів, які необхідно передбачити. Перевагою використання штучних нейронних мереж для виявлення облич є можливість навчання системи для виділення ключових характеристик облич на навчальних вибірках.

Нейронні мережі забезпечують можливість одержання класифікатора добре моделюючого складну функцію розподілу зображень облич. У завданнях класифікації при цьому відбувається неявне виділення ключових ознак усередині мережі, визначення значимості ознак і системи взаємних залежностей між ними. Серед нейронних мереж для розв'язку завдань розпізнавання образів найчастіше застосовуються багатошарові персептрони зі зворотним поширенням помилки, мережі з радіальнобазисною функцією й згорточні нейронні мережі (Convolutional Neural Networks) [17].

## 1.3 Аналіз сучасних алгоритмів розпізнавання емоцій

### 1.3.1 Гістограма спрямованих градієнтів

Гістограма спрямованих градієнтів (англ. Histogram of Oriented Gradients, HOG) - дескриптори особливих точок, які використовуються в комп'ютерному зорі і обробці зображень з метою розпізнавання об'єктів. Дана техніка заснована на підрахунку кількості напрямків градієнта в локальних областях зображення. Цей метод схожий на гістограми напрямки краю, дескриптори SIFT і контексти форми, але відрізняється тим, що обчислюється на щільної сітці рівномірно розподілених осередків і використовує нормалізацію перекривати локального контрасту для збільшення точності.

Основною ідеєю алгоритму є припущення, що зовнішній вигляд і форма об'єкта на ділянці зображення можуть бути описані розподілом градієнтів інтенсивності або напрямком країв. Реалізація цих дескрипторів може бути проведена шляхом поділу зображення на маленькі пов’язані області, іменовані осередками, і розрахунком для кожного осередку гістограми напрямків градієнтів або напрямків країв для пікселів, що знаходяться всередині осередку. Комбінація цих гістограм і є дескриптором.

Для збільшення точності локальні гістограми піддаються нормалізації по контрасту. З цією метою обчислюється міра інтенсивності на більшій фрагменті зображення, який називається блоком, і отримане значення використовується для нормалізації. Нормалізовані дескриптори мають кращу інваріантність по відношенню до висвітлення [18].

Дескриптор HOG має кілька переваг над іншими дескрипторами. Оскільки HOG працює локально, метод підтримує інваріантність геометричних і фотометричних перетворень, за винятком орієнтації об'єкта. Подібні зміни з'являться тільки в великих фрагментах зображення. Більш того, грубе розбиття простору, точне обчислення напрямків і сильна локальна фотометрична нормалізація дозволяють ігнорувати руху пішоходів, якщо вони підтримують вертикальне положення тіла. Дескриптор HOG, таким чином, є хорошим засобом знаходження облич на зображеннях.

Першим кроком обчислень у багатьох детекторах особливих точок є нормалізація кольору і гамма-корекція. Дослідження показали, що для дескриптора HOG цей крок можна опустити, оскільки подальша нормалізація дасть той же результат. Тому на першому етапі розраховуються значення градієнтів. Найпоширенішим методом є застосування одновимірної диференційної маски в горизонтальному або вертикальному напрямку. Цей метод вимагає фільтрації колірної або складової яскравості за допомогою наступних фільтруючих ядер:

$$[-1, 0, 1] \text{ і } [-1, 0, 1]^T$$

Використання більш складних масок, такі як Собел 3x3 (Оператор Собеля) або діагональні маски показали нижчу продуктивність для даного завдання [19]. Використання розмивання по Гаусу перед застосуванням диференціальної маски, зменшує швидкодію без помітного покращення якості.

На наступному кроці обчислюються гістограми осередків. Кожен піксель в осередку бере участь в підвішеному голосуванні для каналів гістограми напрямків, заснованому на значенні градієнтів. Осередки можуть бути прямокутної або круглої форми, канали гістограми рівномірно розподіляються від 0 до 180 або ж від 0 до 360 градусів, в залежності від того, обчислюється «знаковий» або «беззнаковий градієнт».

Для прийняття до уваги яскравості і контрастності градієнти локально нормалізують, для чого осередки згруповують в більш великі зв'язуючі блоки. Дескриптор HOG, таким чином, є вектором компонент нормованих гістограм осередків з усіх областей блоку. Як правило, блоки перекриваються, тобто кожен осередок входить більш ніж в один кінцевий дескриптор. Використовуються дві основні геометрії блоку: прямокутні R-HOG і круглі C-HOG. Блоки R-HOG зазвичай є квадратними сітками, що характеризуються трьома параметрами: кількістю осередків на блок, кількістю пікселів на осередок і кількістю каналів на гістограму осередки.

Таблиця 1.2 – Прівняння дескрипторів на різних наборах зображеннь

| Дескриптор | Набір зображень | Частка пропущених Зображень | Частка помилок першого роду |
|---|---|---|---|
| HOG | MIT | ≈0 | 10−4 |
| HOG | INRIA | 0.1 | 10−4 |
| Вейвлети Хаара | MIT | 0.01 | 10−4 |
| Вейвлети Хаара | INRIA | 0.3 | 10−4 |
| PCA-SIFT, контексти форми | MIT | 0.1 | 10−4 |
| PCA-SIFT, контексти форми | INRIA | 0.5 | 10−4 |

Дослідження проводилися на двох різних наборах даних. База даних пішоходів Мессачуссетського технологічного інституту містить навчальну вибірку з 509 зображень і тестову вибірку з 200 зображень. Набір містить зображення людей тільки спереду або ззаду, пози на зображеннях майже не відрізняються. Ця база даних широко відома і використовується в різних дослідженнях.

Другий набір даних був спеціально створений авторами методу, оскільки на наборі MIT дескриптори HOG показали майже досконалі результати. Цей набір даних, відомий як INRIA, містить 1805 зображень людей. Набір містить зображення людей в широкому розмаїтті поз, включає в себе зображення з важким фоном (наприклад, на тлі натовпу), і є набагато більш складним для розпізнавання [20].

### 1.3.2 Нейронна мережа

Наявна кількість різноманітних підходів класифікації образів вимагає окреслення переваг використання саме багатошарового персептрону. Як нейронна мережа персептрон характеризується такими властивостями, як “Відображення вхідної інформації у вихідну”, “Адаптивність”, “Очевидність відповіді”, “Відмовостійкість”, “Ефективність реалізації на НВІС”, “Аналогія з нейробіологією” та інші . На рис. 1.2 наведено архітектуру персептрону. Для навчання багатошарового персептрону використовується алгоритм зворотного поширення помилки .

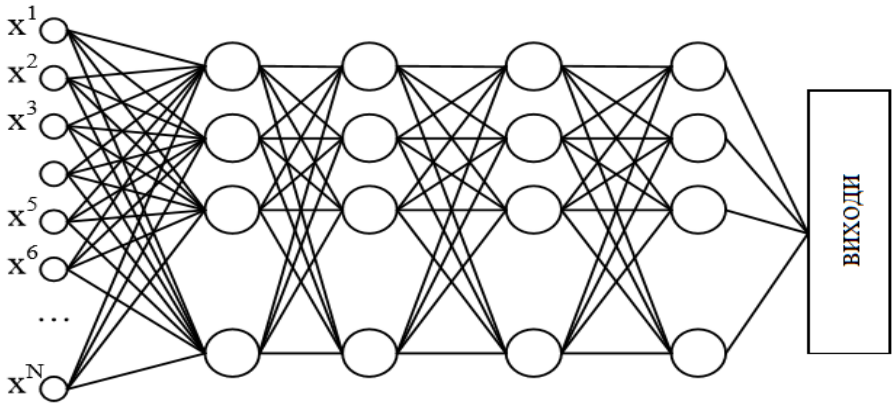


Рисунок 1.2 – Архітектура багатошарового персептрону

Архітектуру системи розпізнавання подано на рисунку 1.3. Архітектура системи складається із шести підсистем, в яких відбуваються певні етапи обробки вхідного зображення (рис. 1.3).

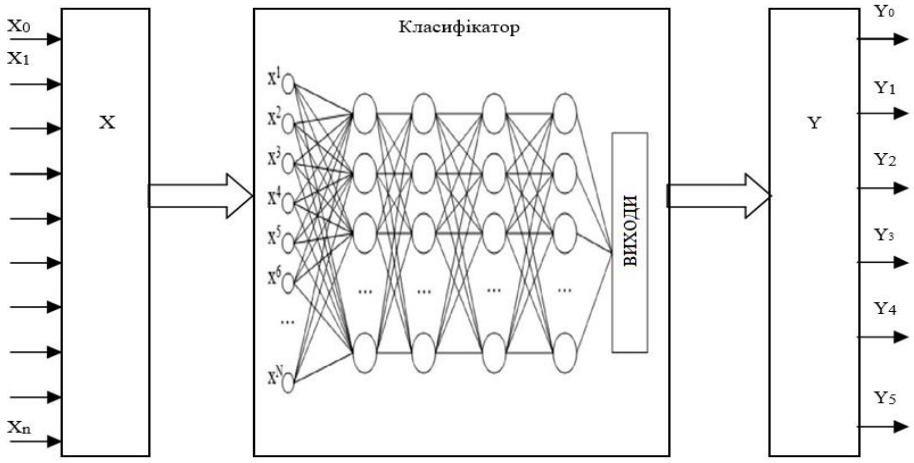


Рисунок 1.3 – Архітектура системи розпізнавання

Підсистема розпізнавання емоцій складається із наступних частин (рис. 1.4):

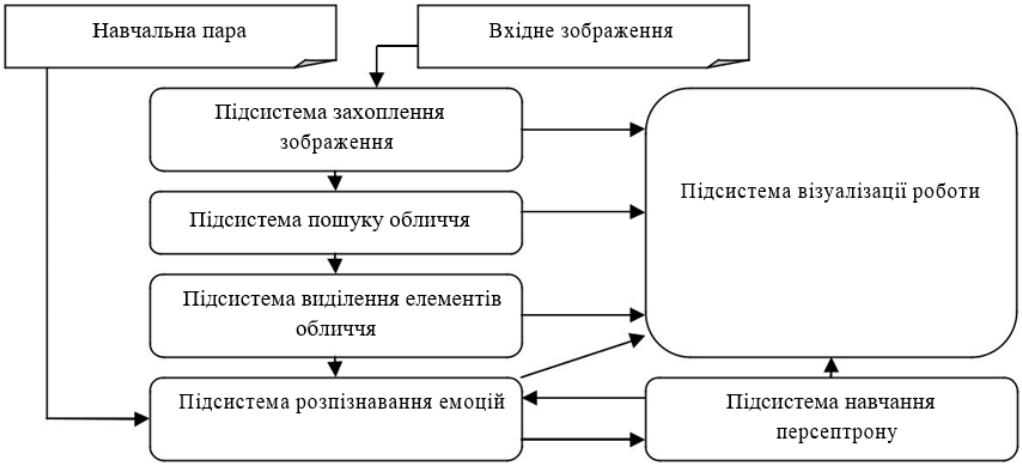


Рисунок 1.4 – Підсистема розпізнавання емоцій

– вхідний образ X, що являє собою відносні координати (x, y) елементів обличчя із початком відліку в центрі обличчя, а також множину сигналів із пороговими значеннями 0 або 1, що відповідають відсутності чи присутності зморщок на чолі, біля очей, на носі, на підборідді. Так корисна інформація потрапляє в нейронну мережу, яка далі класифікує розпізнану емоцію;

– класифікатор на основі багатошарового персептрону. Кількість шарів персептрону змінюватиметься від 1 до 3. Вихідний шар містить 6 нейронів, які відповідають за кожну базову емоцію. Значення вихідних сигналів у межах [0, 1], де 0 означає, що ознак базової емоції не знайдено, а 1 – що базова емоція присутня на зображенні у найбільш явному вигляді (чітко виражена);

– вихідний сигнал – це значення вихідного вектора персептрону, що відповідають за кожну базову емоцію: гнів, відраза, страх, радість, сум, здивування.

Передові дослідження в галузі психології довели, що емоції всіх людей мають спільні зовнішні прояви у міміці людей. Це дає підстави розглядати можливість створення універсального класифікатора емоцій. У роботі запропоновано підхід до розпізнавання мімічних проявів емоцій людини із використанням багатошарового персептрону, який має такі переваги:

– висока точність розпізнавання;

– висока швидкодія;

– адаптація до змін вхідних образів;

– низький рівень споживання ресурсів.

Однак лише за мімічними проявами неможливо із 100% ймовірністю виявити присутність тієї чи іншої емоції. Для вирішення цієї проблеми пропонується інтегрувати цей класифікатор у систему поряд із класифікаторами, які аналізують голос, пантоміміку, зміни фізіологічних процесів в організмі тощо.

### 1.3.3 Приховані моделі Маркова

Одним з статистичних методів розпізнавання осіб є приховані моделі Маркова (ПММ) з дискретним часом. ПММ використовують статистичні властивості сигналів і враховують безпосередньо їх просторові характеристики. Елементами моделі є: множина прихованих станів, множина спостережуваних станів, матриця перехідних ймовірностей, початкова ймовірність станів. Кожному відповідає своя модель Маркова. При розпізнаванні об'єкта перевіряються згенеровані для заданої бази об'єктів моделей Маркова і шукається максимальна із спостережуваних ймовірність того, що послідовність спостережень для даного об'єкта згенерована відповідною моделлю.

Для вирішення цього завдання необхідно виконати два етапи: виявити обличчя на фотографії і розпізнати його. Елементами ПММ є множина спостережуваних символів (спостереження в даному випадку - це коефіцієнти дискретного косинусного перетворення фотографії обличчя людини скануючим вікном певного розміру), множина різних станів, набір спостережуваних символів, вектор початкових станів, матриця перехідних ймовірностей, матриця ймовірностей спостережуваних символів. Можливість застосування ПММ до задачі розпізнавання осіб пов'язано з тим, що:

– в наявної послідовності псевдовипадкових змінних (фрагментів обличчя) ми можемо вважати, що кожне спостереження є незалежним від попередніх;

– кожна випадкова змінна дає вимір, розподіл ймовірностей яких залежить від стану.

Завдання розпізнавання ділитися на дві підзадачі: на етапі навчання ПММ треба побудувати модель по набору різних зображень особи конкретної людини, що

зберігаються в базі даних; на етапі розпізнавання деяке запропоноване зображення слід віднести до однієї з моделі з будь-якої ймовірністю (рис. 1.5).

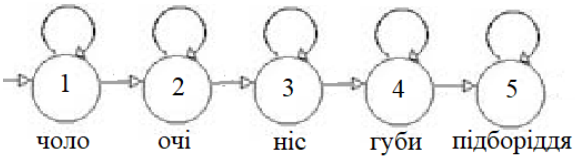


Рисунок 1.5 – Приклад марківського ланцюга одновимірної ПММ.

На базі даних Olivetti Research Ltd був досягнутий середній відсоток розпізнавання близько 90%, максимальний відсоток розпізнавання склав 95%. ORL - база даних, що складається з 400 напівтонових зображень 40 людей - співробітників Olivetti Research Ltd і студентів Кембриджського університету.

Деякі співвідношення параметрів моделі призводять до збільшення відсотка розпізнавання (наприклад, при 85% перетину параметрів скануючого вікна). Ускладнення моделі певним компромісним варіантом між одновимірною ПММ і умовно двовимірною ПММ реалізується зменшенням скануючого вікна по ширині і просуванням його по змієподібній траєкторії: зліва на право і зверху вниз (рис. 1.6). [21]

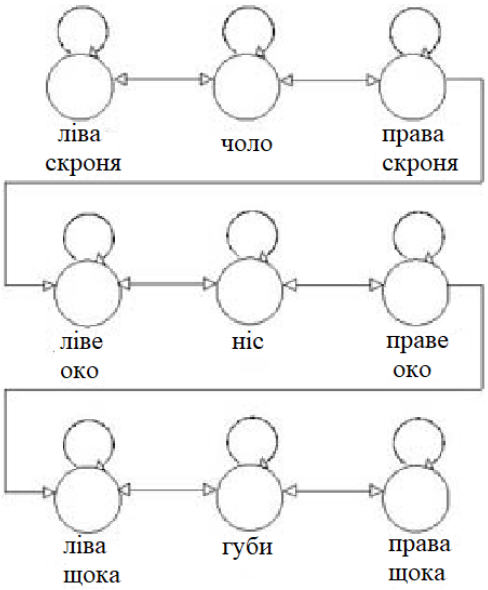


Рисунок 1.6 – Приклад марківського ланцюга ускладненої одномірної ПММ

На тій же базі даних було досягнуто середній відсоток розпізнавання в 85% і максимальний в 96.5%. Падіння числа вірно розпізнаних зображень (в середньому)

пов'язано, по всій ймовірності з тим, що в результаті роботи алгоритму визначення оптимальної послідовності станів відбувалося тільки в горизонтальному напрямку, а у вертикальному напрямку послідовності станів ініціювалася тільки на початковому етапі.

### 1.3.4 Метод на основі положення ключових точок

Для адекватної оцінки впливу інтенсивності пікселів в області обличчя на точність розпізнавання емоцій необхідно в першу чергу визначити базовий алгоритм, з яким буде проводитися порівняння. У даній роботі в якості базового використовується один з варіантів класичного алгоритму на основі положення ключових точок. По суті, цей алгоритм розділений на два етапи: отримання координат ключових точок і класифікація емоцій на їх основі. Класифікація об'єктів по вектору ознак (в даному випадку координат точок) є стандартною задачею навчання з учителем і не представляє інтересу. Слід лише зазначити, що використовувався один з найбільш популярних класифікаторів, а саме метод опорних векторів (англ. Support vector machines, SVM). Отримання ключових точок, однак, є набагато більш складним завданням (Рис. 1.7).

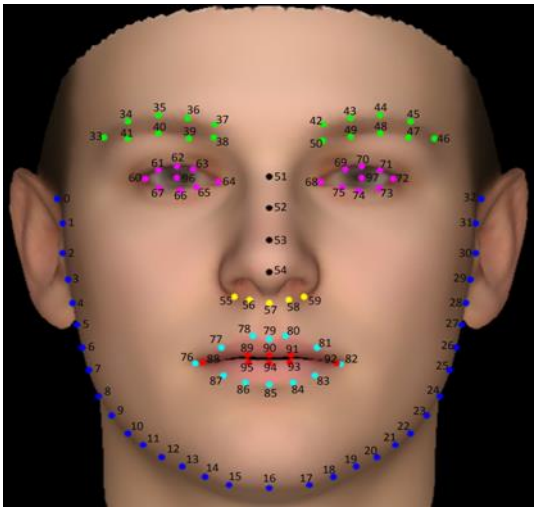

Рисунок 1.7 – Використання ключових точок для опису контуру основних елементів обличчя

Найбільш популярний метод вткористовується у так званих активні моделі зовнішнього вигляду (англ. Active appearance models, AAM) [22]. Вхідними даними для

методу є набір зображень обличь, в якому кожному зображенню відповідає файл розмітки, що містить координати ключових точок, обраних людиною. За цими даними AAM будує дві статистичні моделі: модель форми - параметричну лінійну модель, що описує можливі варіації положення ключових точок. Формою при цьому називається вектор координат ключових точок - подібну модель, але описує вже можливі варіації інтенсивності пікселів. Відповідно текстурою називається вектор всіх пікселів всередині зовнішнього контуру форми [23].

В цьому підході інформація про текстуру використовується для отримання координат ключових точок, але на етап класифікації вона не передається.

### 1.3.5 Метод на основі текстурного аналізу

Можливість визначення емоцій за матеріальним становищем і формі ключових елементів обличчя добре вивчена як психологами, так і фахівцями в галузі інформатики. У той же час аналогічні судження щодо текстури практично не робилися. Існує можливість визначення (вручну або комп'ютерними засобами) виразу обличчя людини навіть за відсутності інформації про становище і формі його ключових елементів. Іншими словами, передбачається, що емоції на обличчі кодуються не тільки положенням ключових точок, а й іншими ознаками (рис. 1.8).


Рисунок 1.8 – Приклади приведення до середньої формі: а) вихідні зображення осіб; б) відповідні зображення, приведені до середньої формі

Всі ключові точки брів, очей, носа, губ, а також підборіддя знаходяться в ідентичному положенні. Дані зображення отримані шляхом тріангуляції по ключовим точками і застосування кусочно-афінного перетворення до знімків осіб, що виражають емоції. Незважаючи на вирівнювання положення ключових елементів особи, емоції на ньому все ще легко вгадуються по зморшках, тіням та інших елементах текстури.

Як і в методі на основі положення ключових точок, перетворення до усередненої форми дозволяє сформувати для кожного зображення відповідний вектор текстури.

Цей вектор можна безпосередньо використовувати в якості вектора ознак при класифікації. В якості класифікатора можна застосовувати вже відомий нам SVM. В результаті отримуємо наступний метод розпізнавання емоцій на основі інформації про текстуру. На етапі навчання:

– зображення піддаються попередній обробці, т. д. За допомогою активних моделей зовнішнього вигляду на них визначаються положення ключових точок, а потім кусочно-афінним перетворенням особи на зображеннях приводяться до усередненої формі;

– пікселі отриманих зображень разом з відповідними позначками емоцій використовуються для навчання методом SVM;

На етапі застосування:

– За аналогією зі стадією навчання нові зображення приводяться до середньої форми;

– Перетворені зображення класифікуються за допомогою навченої раніше моделі. Забігаючи наперед, скажемо, що точність розпізнавання методом на основі текстури лише трохи поступається точності класичного підходу.

Підводячи проміжний підсумок, підкреслимо ще раз спільні та відмінні риси двох методів. Обидва включають етапи навчання (створення моделі) і застосування. Обидва вимагають попередньої обробки зображень для отримання ознак. Перший метод використовує в якості ознак координати ключових точок (форму), отриманих за допомогою моделей активного способу, другий - текстуру зображення всередині зовнішнього контуру обличчя. В якості кінцевого класифікатора в обох описаних

методах використовується SVM. Для приведення обличь до єдиної формі застосовується активна модель зовнішнього вигляду, проте це не є обов'язковою вимогою. Для аналогічних цілей може бути використаний пошук кількох ключових точок і подальше 3D-моделювання [24].

### 1.3.6 Комбінований метод

Хоча описані вище методи дають досить високі результати самі по собі, можливо об'єднати їх, поєднавши обидві набори ознак: координати ключових точок і інтенсивності пікселів в області обличчя. Оскільки теоретичне обгрунтування для всіх частин алгоритму вже було дано в попередніх розділах, тут просто опишемо послідовність кроків комбінованого методу розпізнавання емоцій. На етапі навчання:

– до вхідних зображень застосовуються активні моделі образу для отримання координат ключових точок. Вектори ключових точок для кожного зображення організовуються в матрицю S;

– до зображень застосовується кусочно-афінне перетворення для приведення їх до середньої форми: для отриманих на попередньому кроці точок будується тріангуляція Делоне, а потім пікселі всередині кожного окремого трикутника відображаються на відповідні пікселі всередині середньої форми.

– пікселі кожного перетвореного зображення організовуються в єдиний вектор ознак, а вектори для всіх зображень організовуються в матрицю T;

– матриці S і T об'єднуються так, що кожен вектор s з матриці S розширюється відповідним вектором із матриці T. Іншими словами, ознаки з двох наборів об'єднуються в єдиний вектор. Результуючу матрицю назвемо X;

– мітки емоцій об'єднуються в єдиний вектор y;

– матриця даних X і вектор міток y використовуються для навчання моделі SVM.

Як і передбачалося, такий комбінований підхід дав точність, що перевершує результати обох описаних раніше методів.

## 1.4 Висновки до першого розділу

Проведено аналіз наукових робіт та сучасних алгоритів, розпізнавання роботом емоцій, на обличчі людини. Для аналізу були відібрані найпопулярніші та найефективніші алгоритми що застосовуються для розпізнавання. Серед них такі як:

– метод на основі текстурного аналізу;
– багатошаровий персептрон;
– алгоритм AdaBoost;
– приховані Марківські моделі.

Порівнюючи показники AAM та SVM, можна сказати, що на стадії застосування AAM визначає положення ключових точок в середньому за 180-190 мс, а SVM проводить класифікацію приблизно за 25 мс, що робить можливим використання розроблених алгоритмів в системах реального часу. Точність розпізнавання визначалася стандартним методом перехресної перевірки. При цьому для навчання класифікатора завжди використовувалося останнє зображення (особа, максимально сильно виражає емоцію) в кожній з послідовностей кадрів (всього 327 зображень).

За результатами аналізу були отримані наступні результати: алгоритм adaboost – 94% правильних відповідей; метод на основі текстурного аналізу – 93% правильних відповідей; метод на основі прихованих Марківських мереж – 92% правильних відповідей, багатошаровий персептрон 94% (рис. 1.9). Таким чином, можемо зробити висновок що найкраще за результатами дослідження пройшли системи на основі багатошарового персептрону та алгоритму adaboost.


Рисунок 1.9 – Результати аналізу

# 2 ДОСЛІДЖЕННЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ЕМОЦІЙ НА ОБЛИЧЧІ ЛЮДИНИ

## 2.1 Дослідження алгоритму AdaBoost.

Алгоритм навчання AdaBoost, запропонований Фрейндом і Шпапіром, в його оригінальній формі, використовується для підвищення класифікаційної ефективності простого алгоритму навчання. Досягається це шляхом поєднання наборів ознак слабких класифікаторів, щоб створити більш міцний класифікатор. Простий алгоритм навчання називається слабким класифікатором. AdaBoost - це не тільки швидкий класифікатор; це також метод вибору особливостей. Перевага вибору ознак AdaBoost полягає в тому, що ознаки вибираються залежно від вже обраних ознак. У дослідженні використовується варіант багатокласовго AdaBoost, який використовується для вибору важливих значущих ознак достатніх для розпізнавання емоцій на обличчіі людини.

Прототипні вектори ознак для кожного класу виразів обличчя створюються шляхом прийняття медіани кожного відповідного елемента у векторах ознак від всього навчального набору цих послідовностей виразів обличчя [28]. Медіана вибирається замість середнього значення, тому що це менше впливає на наявність дисперсії.

Нехай U являє собою базу даних виразів обличчя, яка містить послідовності виразів обличчя. База даних об'єднана в шість різних класів: $U_c$, c = 1, ..., 6, кожен з яких представляє один із шести основних виразів обличчя (гнів, відраза, страх, щастя, смуток та здивування). Припустимо, що прототип вектору признаків класу *c* для векторних признаків типу 1 та векорів признаків типу 2 позначає зміну відстані та кута між i-ї та j-ої парними орієнтирами в 1-му кадрі по відношенню до першого кадру в k-ій відеозйомці.

Метою є пошук невеликої кількості найбільш значущих векторів признаків з усього набору признаків. Для досягнення цієї мети використовується слабкий класифікатор призначений для вибору єдиного вектору признаків, який найкраще класифікує навчальні дані. Рівень класу векторного елемента вирішується на основі мінімальної відстані подібності алгоритму DTW, із векторами прототипів. DTW - відомий алгоритм, який має на меті порівняти та вирівнювати дві послідовності точок даних [29].

Хоча DTW спочатку був розроблений для розпізнавання мови, він також застосовувався для багатьох інших областях. У системі DTW є ефективним способом пошуку подібності між векторами ознак, оскільки довжина вектору можливостей може бути різною за кількістю кадрів у послідовності виразів обличчя, а також у різних осіб, оскільки рух знакових позицій є нелінійним, оскільки вираз обличчя розвивається. Алгоритм DTW використовується щоб швидко знаходити подібність між двома послідовностями. Слабкий класифікатор $(T,(x,x^p,\ f))$ складається з вектора функції (f), вхідного виразу обличчя (x) та прототипних виразів обличчя $(x^p)$:

$$\left(T(x,x^{(p)},f\right) = arg\min_c\left\{d_{DTW}\left(f(x),f(x_c^{(p)})\right)\right\} \quad (2.1)$$

На практиці жодна функція не може виконати цю класифікаційну задачу з низькою помилкою. Особливості, які вибираються в ранньому процесі, призводять до зменшення частоти помилки класифікації, ніж функції, виділені в подальших раундах. Нище представлено оригінальний алгоритм багатокласового AdaBoost.

1. Установка ваг $w_{1,i} = 1/n,\ i = 1,2,\dots,n$;

2. Для m = 1 до M:

a. Нормалація ваги $w_{m,i} \leftarrow w_{m,i}/\sum_{j=1}^{n} w_{m,j}$

b. Вибір найкращого класифікатору відносно вагової помилки:

$$err^m = min_f \sum_{i=1}^{n} w_i I\left(c_i \neq T(x_i,x^p,f)\right)/\sum_{i=1}^{n} w_i \quad (2.2)$$

c. Визначення $T^m(x) = T(x,x^p,f_m)$,

де, $f_m$ це мінімум помилки $err^{(m)}$;

d. Обчислення:

$$a^m = \lg\frac{1-err^{(m)}}{err^{(m)}} + \lg(K-1) \quad (2.3)$$

e. Оновлюємо ваги:

$$w_{m,i} \leftarrow w_{m,i}\ exp(\alpha^m I(c_i \neq T^m(x_i)), i = 1,\dots,n$$

3. Отримання остаточного класифікатора:

$$C(x) = arg\max_k \sum_{m=1}^{M} a^{(m)} I(T^m(x) = k) \quad (2.4)$$

Базовий алгоритм AdaBoost:

На вході: $(x_1,y_1),...,(x_m,y_m)$; $x_i \in X, y_i \in \{-1,1\}$;

Задаємо ваги: $D_1(i) = \frac{1}{m}$;

Для t = 1,...,T:

1. Знаходимо $h_t = \arg\min_{h_j \in H} e_j$;

$$e_j \sum_{i=1}^{m} D_t(i) I\left[y_i \neq h_j(x_i)\right] \quad (2.5)$$

2. Якщо $e_t \geq \frac{1}{2}$ тоді зупиняємо пошук;

3. Установка значення $\alpha_t = \frac{1}{2} log \frac{1+e_t}{e_t}$

4. $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

На виході отримуємо фінальний класифікатор:

$$H(x) = sign \sum_{t=1}^{T} \alpha_t h_t(x) \quad (2.6)$$

Виділяють дві ключові властивості AdaBoost. По-перше, як було показано алгоритм мінімізує верхню межу на похибці класифікації $\varepsilon_{tr}(H)$ на тренувальному наборі:

$$\varepsilon_{tr}(H) \leq \prod_{t=1}^{T} Z_t = \frac{1}{2^T} \prod_{t=1}^{T} \sqrt{e_t(1-e_t)} \quad (2.7)$$

Ця верхня межа мінімізується, вибравши слабкий класифікатор з найменшою зваженою помилкою на тренувальному наборі, виконаному на кроці 1, і встановивши його коефіцієнт, як це було зроблено на етапі 3. По-друге, схема повторного вагування гарантує, що оновлений розподіл задовольняє:

$$\sum_{i=1}^{m} D_{t+1}(i) u_{t,i} = 0, \quad (2.8)$$

де $u_{t,i} = h_t(x_i) y_i$. Після перезапису вагової помилки на кроці 1 алгоритму AdaBoost до еквівалентної форми

$$h_{t+1} = arg \max_{h_q \in H} \sum_{i=1}^{m} D_{t+1}(i) u_{t,i} \quad (2.9)$$

очевидно, що обраний $h_{t+1}$ є "максимально незалежним" від помилок, які присутні в h з $u_{t,i}$. Відзначимо також, що рівняння (9) має наслідки для зваженої похибки $e_t^{t+1}$ з $h_t$, де верхній індекс вказує на те, що похибка вимірюється на вагах на етапі t + 1. Оскільки

помилка може бути виражена як

$$e_t^{t+1}\frac{1}{2}(1-\sum_{i=1}^{m}D_{t+1}(i)u_{t,i}) \quad (2.10)$$

з рівняння (2.9) помилка t + 1 t = 1/2. Таким чином, слабкий класифікатор $h_t$ еквівалентний випадковому припущенню під час t + 1. Узагальнюючи рівняння (2.8) – (2.10), алгоритм AdaBoost мінімізує верхню межу похибки класифікації, вибирає слабкі класифікатори з найменшою помилкою, а слабкий класифікатор, вибраний в момент часу t, максимально незалежно від слабкого класифікатора, вибраного в момент t - 1. Ці властивості будемо використовувати в подальшому.

## 2.2 Побудова груп класифікаторів для алгоритму AdaBoost

Кожна слабка класифікація покладається на одну функцію (задану позицією в кадрі з обличчям, її розмір і тип). Параметри та порогові значення є параметрами кожного слабкого класіфікатора, і вони повинні бути перераховані на кожному циклі AdaBoost, щоб мінімізувати кількість неправильно класифікованих зразків, зважених поточним розподілом [30].

Пропонується замість того, щоб навчати один класіфікатор, будуємо каскад класифікаторів. При використанні набору класифікаторів, якщо вікно зображення (регіон) передано першому класифікатору класифікується як не обличчя, або рішення відкладається, то зображення передається іншому класифікатору. Метою кожного класифікатора є опущення навчального набору для класу фази наступного етапу каскаду. Оскільки легко розпізнавані набори емоцій на зображеннях класифікуються на ранніх стадіях, класифікатори пізніх стадій каскаду можуть бути навчені швидше лише на більш складній, але меншій частині навчального набору обличчь. Побудова каскаду описується в алгоритмі 2. На вході алгоритму задаємо бажаний рівень похибки f, рівень виявлення d каскадних етапів і відсоток помилково розпізнаних елементів. Кожен етап тренується до досягнення f та d. Оскільки AdaBoost не призначений для досягнення низьких помилкових показників і високих показників виявлення, порогові значення коригуються заздалегідь. Рівень помилок f зазвичай встановлюється на більш високі значення. Збільшення помилок на кроці 2 гарантує експоненціальне зменшення загальної кількості

неправильно розпізнаних елементів. Рівень правильного виявлення повинен бути встановлений близько до одиниці, щоб забезпечити високий рівень фінального розпізнавання D.

Каскадне оцінювання еквівалентно послідовній класифікації з використанням зміненого дерева рішень. Коли на якомусь із етапу позначається регіон у зображенні як не обличчя, процес прийняття рішення закінчується. В іншому випадку запускається класифікатор наступного етапу. Регіон оголошується обличчям, якщо воно прийнято всіма класифікаторами в каскаді. Виявлення обличчя здійснюється шляхом переміщення каскаду по всьому зображенню в декількох масштабах та місцях [31]. Типовий образ містить лише невелику кількість регіонів обличчя в порівнянні з кількістю сканованих регіонів.

У зв'язку з достроковим припиненням процесу прийняття рішень в регіонах без обличчя в середньому оцінюються лише кілька стадій каскаду. Отже, швидкість оцінювання значною мірою залежить від обчислюваної складності та відхилень перших кількох стадій. Запропоновано розширення до алгоритму навчання AdaBoost що допоможе знизити рівень помилкового розпізнавання на перших стадіях, що в кінцевому результаті допоможе підвищити точність розпізнавання всієї системи в цілому.

На вході: Задаємо допустимий рівень похибки $f$, розпізнавання d та $f_{\text{final.}}$

$F_0 = 1,\ D = 1$

$D_0$ until $F$i $> f_{\text{final}}$

1. Тренуємо класифікатор поки$f_{\text{reached}} < f$ and $d_{\text{reached}} > d$

2. $F_{\text{i+1}} = F_{\text{i}} \times f_{\text{reached}}$

3. $D_{\text{i+1}} = D_{\text{i}} \times d_{\text{reached}}$

4. Не враховуємо неправильно класифіковані обличчя та генеруємо нові дані для непов’язаних зображень без обличь.

## 2.3 Запропоновона методика на основі алгоритму AdaBoost

Запропонована методика, відрізняється від стандартного AdaBoost у двох основних аспектах. По-перше, у стандартному AdaBoost, щойно доданий слабкий клас може бути

показаний "незалежним" у точно визначеному способі раніше доданого слабкого класу. Алгоритм TCA знаходить новий слабкий клас, який є незалежно від усіх виділених поки що слабких класів. По-друге, коефіцієнти вже знайдених слабких класів повторюються під час навчального процесу. Запропоновані модифікації мінімізують верхню границю похибки класифікації більш правильніше і знаходять більш короткі набори класифікаторів. Стандартний алгоритм Ківінена та Вармута не оновлював коефіцієнти слабких класифікаторів належним чином. Таким чином, алгоритм втратив важливу властивість мінімізації верхньої межі помилки навчання [32].

У випадку, якщо слабкі класифікатори безпосередньо відповідають особливостям, як у рамці виявлення обличчя Віола та Джонса, зміна одного коефіцієнта на ненульове значення ефективно вибирає цю функцію. Лі та ін. запропонований FloatBoost, модифікація AdaBoost, де деякі з вже ненульових коефіцієнтів встановлюються до нуля, коли вони призводять до нижньої верхньої межі помилки класифікації. Замість жадібного вибору функції використовується метод послідовного переходу вперед (SFFS). Лі та ін. показують, що ця модифікація призводить до скорочення класів. Основним внеском цього розділу є (1) модифікація алгоритму AdaBoost, що призводить до скорочення класифікаторів та прискорення класифікації, (2) впровадження повністю корегувального алгоритму тренування визначення обличчя.

## 2.4 Побудова класифікатора для запропонованої методики

Властивість незалежності, про яку йшла мова, дуже приваблива з точки зору вибору об'єкта. Виникає питання, чи можна знайти новий слабкий класифікатор, максимально незалежний від усіх вже обраних. У такому випадку розподіл $D_{t+1}$ повинен задовольняти:

$$\sum_{i=1}^{m} D_{t+1}(i)u_{t,i} = 0 \text{ для } q = 1, \dots, t, \quad (2.11)$$

де $u_{t,i} = h_q(x_i)y_i$. У системі рівнянь (12) не існує замкнутої форми, а іноді точне рішення навіть не існує [9]. Це є наслідком обмеження невід'ємності на $D_{t+1}$, що є розподілом. Тому TCS розроблений як ітеративний алгоритм оптимізації.

Алгоритм 3 Редагований алгоритм із оновленими коефіцієнтами (TCA: Totally Corrective Algorithm)

На вході: $(x_1,y_1),...,(x_m,y_m)$; $x_i \in X, y_i \in \{-1,1\}$

Ініціалізація ваг: $D_1(i) = \frac{1}{m}$

Для $t = 1,...,T$:

1. Знаходимо $h_t = arg \min_{h_j \in H} e_j$;;

$$e_j \sum_{i=1}^{m} D_t(i) I\left[y_i \neq h_j(x_i)\right] \quad (2.12)$$

2. Якщо $h_t \geq \frac{1}{2}$ тоді зупиняємось

3. Установлення значення $\alpha_t = \frac{1}{2} \log \frac{1+e_t}{e_t}$

4. Оновлення

$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i)}{Z_t}$

5. Вносимо зміни до алгоритму (дивимось Алгоритм 4)

На виході отримуємо фінальний класифікатор:

$$H(x) = sign \sum_{t=1}^{T} \alpha_t h_t(x) \quad (2.13)$$

Алгоритм 4:

Ініціалізація $\hat{D}_0 = D_t$;

Для r j = 1,2,...,$J_{max}$;

1. $q_j = arg\max_{q=1..t} \left|e_q - \frac{1}{2}\right|$;

2. Якщо $\left|e_{qj} - \frac{1}{2}\right| < \Delta_{min}$ виходимо з циклу.;

3. Нехай $\hat{\alpha} = \frac{1}{2} \ln \frac{1-e_{qj}}{e_{gj}}$;

4. Перезважування:

$$D_{j+1}(i) = \frac{1}{Z_j} D_j(i) \exp(-\alpha_j u_{qj,i}) \quad (2.14)$$

5. $\alpha qj = \alpha_{qj} + \alpha_j$

Установлюємо $D_{t+1} = \hat{D}_j$

Класифікатори були перевірені на наборі даних MIT + CMU . Основна мета дослідження полягає в тому, щоб продемонструвати скорочення порівнянні з класичним

підходом Віола-Джонса, а не покращенням швидкості виявлення як такого. Це означає, що ми не намагалися знайти, наприклад, оптимальні набори слабких класифікаторів, оскільки це не важливо для справедливого порівняння AdaBoost і TCA.

Результати для каскадів, навчених двома варіантами AdaBoost, наведені у таблиці 2.1. Для кожного числа етапів в каскаді записуються наступні значення (зліва направо в таблиці 2.1): кількість слабких класифікаторів, що формують етап у каскад, загальна кількість оцінок на кожному етапі, а також кількість помилок першого та другого роду в наборі даних MIT + CMU. Перший рядок, "0-й етап", додається лише для відображення загальної кількості відсканованих регіонів.

Можна помітити, що для обох алгоритмів складність стадій поступово зростає, крім двох невеликих частин. На початку зростання в запропонованому TCA відбувається повільніше, і воно змінюється після чотирьох етапів. Проте складність не є єдиним важливим фактором, що визначає швидкість виявлення обличчя. Також важливо відзначити кількість регіонів, позначених як потенційна особа на кожному етапі. Видно, що TCA відкидає набагато більше регіонів на ранніх стадіях, ніж AdaBoost. Це перше зрізування впливає на показники помилок першого та другого роду, які показані в останніх двох колонках. Ці два показники вимірюють продуктивність каскадного класифікатора. З таблиці видно, що обидва алгоритми мають подібні показники, помилок першого та другого роду але TCA сходяться набагато швидше.

Для порівняння швидкості каскадів, підготовлених TCA та AdaBoost, було виміряно кількість слабких класифікаторів, які оцінювалися на наборі даних MIT + CMU. Усі регіони мають оцінюватися за класом першого етапу. Кількість оцінок, отже, є результатом кількості регіонів та довжини класі першого етапу. Те саме стосується другого (і вищого) класу, але тільки ті регіони, які не відхилені на першому (попередньому) етапі, оцінюються. Підсумовування оцінок номерів першого та другого етапів дає кількість оцінок  класифікатора каскадів (рис. 2.1).

Складність каскадного класифікатора зі співставленням помилок першого та другого роду  в чотири рази менше для алгоритму TCA (шестиступенева TCA проти одинадцятиступінчастої AdaBoost). Кількість оцінок, необхідних у AdaBoost, на 20% вище, ніж у TCA.

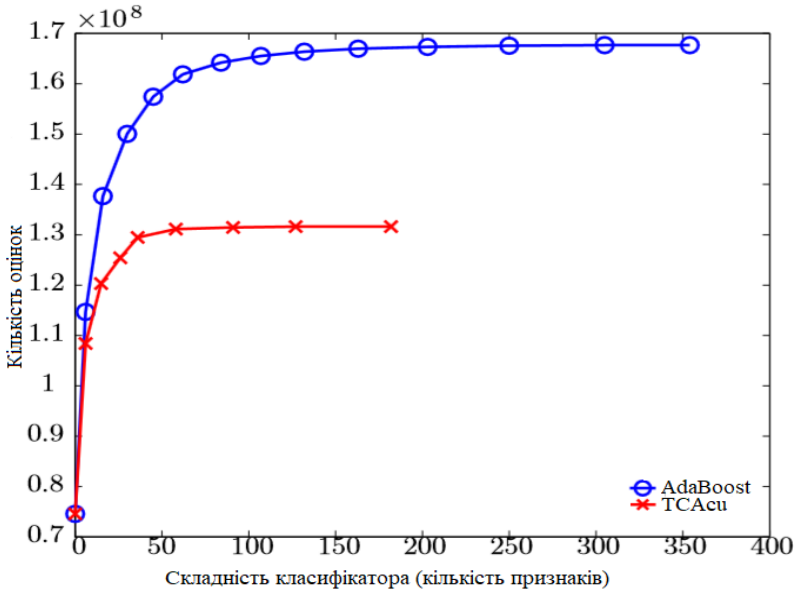


Рисунок 2.1 – Порівняння вибірок Adaboost та TCA

## 2.5 Навчання алгоритму TCA

Під час тренувального процесу, дані для кожної із стадій тренування та перевірки оновлюються (див. Алгоритм 2). Номера зображень набору даних для навчання та валідації складається з 5000 випадково відібраних регіонів із пронумерованих зображень.

Включені лише регіони, які не були відхилені попередніми стадіями каскадного класифікатора. Обличчя, відкинуті деякими стадіями класифікації, відкидаються, але каскад будується, для забезпечення того щоб ці помилкові відхилення становили лише невелику частину даних обличчя. У представлених експериментах значення були встановлені 0,999 для правильно відкунутих обличчь та 0,0001 для хибно відкинутих наборів обличчь, тобто помилок першого та другого роду.

Остаточний, позитивний показник був досягнутий на восьмому етапі в TCA та на десятому етапі AdaBoost. На рисунку 2.1 представлено порівняння вибірок Adaboost та TCA. Горизонтальна вісь: складність каскадного класифікатора, виражена числом слабких класіфікаторів U.

Алгоритм AdaBoost широко використовувався для визначення підмножини гістограми LBP для кожного виразу обличчя. Оскільки кожна гістограма LBP розраховується з субрегіону, AdaBoost фактично використовується для пошуку субрегіонів, які містять більше значущої інформації для вираження емоції.

Для визначення функцій, важливих для дискримінації одного конкретного класу виразів обличчя, необхідно знайти підмножину вектора функцій для кожного виразу обличчя.

Таблиця 2.1 – Порівняння продуктивності роботи алгоритмів AdaBoost та TCA

| № | Кількість ознак класифікатора | | Кількість оцінок | | Похибка першого роду | |
|---|---|---|---|---|---|---|
| | AB | TCA | AB | TCA | AB | TCA |
| 0 | 0 | 0 | 12431151 | 12431151 | | |
| 1 | 6 | 6 | 4009205 | 3757632 | 0 | 0 |
| 2 | 10 | 9 | 1643072 | 1083123 | 0 | 0 |
| 3 | 14 | 11 | 823246 | 512004 | 0 | 1 |
| 4 | 15 | 10 | 435483 | 183962 | 2 | 5 |
| 5 | 17 | 22 | 201982 | 49814 | 4 | 16 |
| 6 | 22 | 33 | 100887 | 9052 | 11 | 39 |
| 7 | 23 | 36 | 52867 | 3173 | 17 | 83 |
| 8 | 25 | 55 | 27504 | 1149 | 26 | 143 |
| 9 | 31 | | 14818 | | 35 | |
| 10 | 40 | | 7568 | | 53 | |
| 11 | 47 | | 4194 | | 59 | |
| 12 | 55 | | 2602 | | 73 | |
| 13 | 49 | | 1828 | | 84 | |

Наприклад, якщо необхідно знайти ознаки гніву виразів обличчя, гнів - це позитивний клас, а решта обличчя (огида, страх, щастя, смуток та здивування) належать до негативного класу. Слабкий класифікатор класифікує вхідний мімічний вираз в позитивний клас, якщо відстань подібності DTW є мінімальною з позитивним класом прототипу серед всіх прототипів; інакше він класифікує вхідний вираз обличчя у негативний клас. На рисунку 2.2 показано перші кілька функцій, вибраних для кожного класу мімічних виразів, використовуючи цю схему. Відзначається, що різні класи вираження обличчя мають різні ключові дискримінаційні геометричні риси [33].

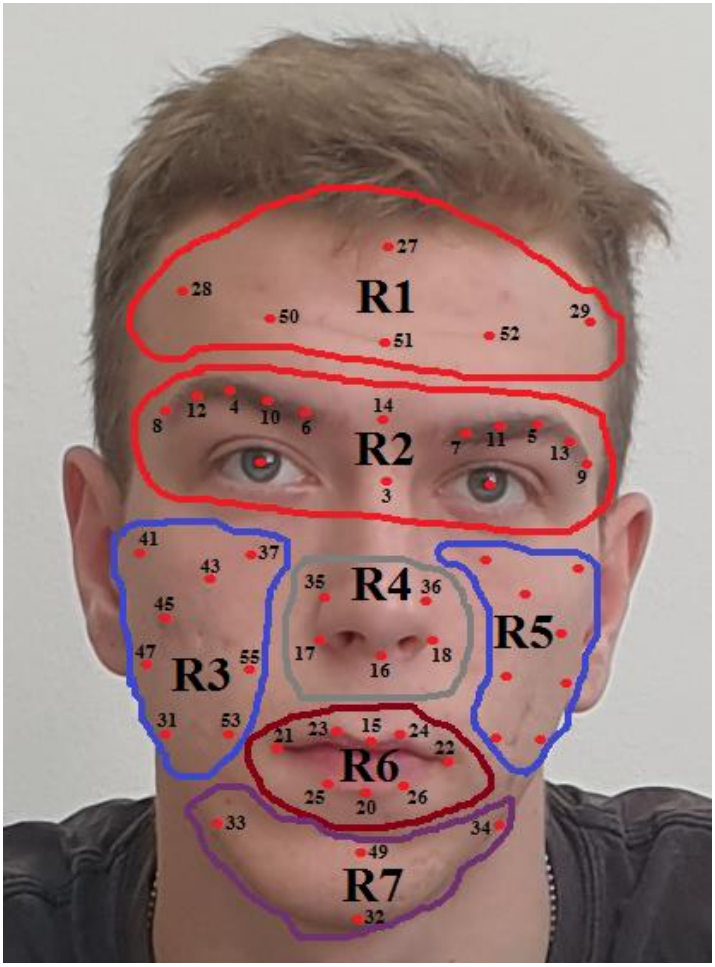


Рисунок 2.2 – Регіони обличчя

Таблиця 2.2 – Важливі регіони для вираження окремої емоції

| Емоція | Регіон обличчя (одиничний/парний) з найбільш значущими ключовими точками |
|---|---|
| Злість | R1-R2, R2, R2-R3, R5-R7, R6, R6-R7 |
| Огида | R1-R2, R2, R2-R3, R5, R5-R6, R5-R7 |
| Страх | R1-R2, R2, R3-R6, R5-R6, R2-R7, R4-R7, R6-R7 |
| Радість | R1-R2, R2, R2-R4, R2-R6, R3-R6, R4-R6 |
| Смуток | R2-R3, R2, R3-R7, R5-R6, R6, R5-R7, R6-R7 |
| Здивування | R1-R3, R2, R2-R3, R2-R7, R2-R5, R7 |

## 2.6 Набори баз даних які використовувались

База даних Extended Cohn-Kanade (CK+) [34] була використана для розпізнавання обличчя в шести основних класах експресії обличчя (гнів, відраза, страх, щастя, смуток та здивування). Ця база даних складається з 593 послідовностей з 123 осіб. Послідовність зображень змінюється залежно від тривалості (тобто від семи до 60 кадрів), і включає

початок (який також є нейтральним обличчям) до пікового формування міміки. Послідовності зображень від нейтрального до цільового дисплея оцифровані в масиви 640 × 480 або 640 × 490 пікселів. Тільки 327 із 593 послідовностей мають певний емоційний клас. Це тому, що це єдині, які відповідають прототипному визначенню. Для нашого дослідження, 315 послідовностей набору даних вибираються з бази даних для базового розпізнавання обличчя.

Точність класифікації – це середня точність у всіх шести випробуваннях. Щоб отримати кращу оцінку точності розпізнавання окремого типу виразів, даються матриці суперечностей. Матриця суперечностей являє собою n × n матрицю, в якій кожен стовпець матриці представляє екземпляри у передбачуваному класі, тоді як кожен рядок являє собою екземпляри у фактичному класі. Діагональні елементи матриці суперечностей – це показники виразів обличчя, які правильно класифікуються, тоді як позадіагональні записи відповідають нестандартним ставкам.

## 2.7 Розпізнавання обличчя за допомогою Multi-Class AdaBoost

У результаті дослідженні запропоновано варіант багатокласного AdaBoost, що використовується для вибору векторів дискримінаційних властивостей, витягнутих з результатів відстеження орієнтирів.

У той же час, алгоритм AdaBoost визначає ваги, пов'язані з кожним вектором властивостей. Мімічні вирази визначаються за допомогою сильного класифікатора, заданого в рівнянні (2.12).

Класифікація базується на відстані подібності DTW (dynamic time warping – що представляє собою техніку вирівнювання часових рядів) між вибраними векторами можливостей послідовності експресії обличчя, з вектором можливостей, пов'язаним з кожним класом прототипної послідовності виразів обличчя. Кожен векторний елемент класифікує вираз обличчя в один із шести класів відповідно до мінімальної відстані подібності DTW. Одним з переваг використання вимірювання подібності DTW є те, що два порівнювані вектори, які потрібно порівнювати, не обов'язково мають однакову довжину. Збільшення кількості функцій в рівнянні також підвищує точність класифікації

до деякої межі. На рисунку 2.3 наведено графік кількості функцій проти точності розпізнавання, як для даних навчання, так і для тестування.

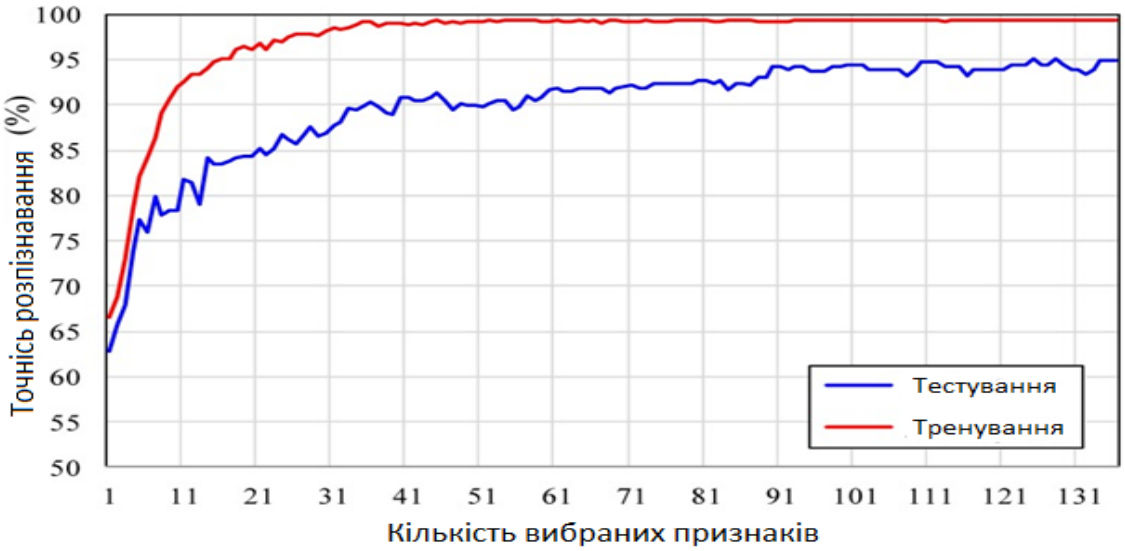


Рисунок 2.3 – Зміна точності розпізнавання відповідно до кількості вибраних ознак класифікатора

В результаті відстеження заданих 52 орієнтирами отримано загалом 1378 можливих векторних функцій, але лише декілька з них достатні для того, щоб виділити шість основних мімічних виразів. Найвища точність класифікації 95,1% досягається принаймні при використанні 125 векторних ознак. Таблиці 2.3 та 2.4 показують матрицю суперечностей розпізнавання виразів обличчя, використовуючи багатокласовий AdaBoost з 75 та 125 множини вектору ознак, відповідно. Деякі емоції, наприклад страх та здивування для алгоритму дуже подібні, тому їх найважче правильно розпізнати.

Таблиця 2.3 – Розпізнавання емоцій використовуючи мульти-класовий AdaBoost в якого 75 векторів признаків

| % | Злість | Огида | Страх | Щастя | Сум | Здивування |
|---|---|---|---|---|---|---|
| Злість | 92.5 | 2.5 | 0 | 0 | 5 | 0 |
| Огида | 0 | 96.6 | 1.6 | 0 | 1.6 | 0 |
| Страх | 0 | 0 | 92 | 8 | 0 | 0 |
| Щастя | 0 | 3 | 1.5 | 95.3 | 0 | 0 |
| Сум | 6.6 | 0 | 3.3 | 0 | 87.6 | 3.3 |
| Здивування | 0 | 0 | 2.5 | 0 | 2.5 | 95 |

Сум та гнів це вирази які інколи буває важко розпізнати навіть для людей, не говорячи вже про машину. Крім того інколи деякі із виразів страху та здивування система

плутала та неправильно розпізнавала, частіше всього страх розпізнавався як здивування.

Таблиця 2.4 – Розпізнавання емоцій використовуючи мульти-класовий AdaBoost в якого 125 векторів признаків

| % | Злість | Огида | Страх | Щастя | Сум | Здивування |
|---|---|---|---|---|---|---|
| Злість | 95 | 5 | 0 | 0 | 0 | 0 |
| Огида | 0 | 95 | 1.6 | 3.3 | 0 | 0 |
| Страх | 0 | 0 | 92 | 8 | 0 | 0 |
| Щастя | 0 | 0 | 3.1 | 96.9 | 0 | 0 |
| Сум | 6.6 | 0 | 0 | 0 | 93.3 | 0 |
| Здивування | 0 | 0 | 0 | 1.2 | 0 | 98.7 |

Жоден інтерактивний векторний елемент не може класифікувати мімічні вирази з високою точністю. Сильний класифікатор  класифікує будь-який вхідний вираз для одного з шести базових класів, який має найвищий показник довіри.

Також потрібно знати для кожного виразу обличчя, який відсоток векторів функцій класифікується правильно, і який відсоток векторів ознак неправильно класифіковано. Це дасть більш точні дані про оцінку кожного виразу обличчя. На рисунку 2.4 показані суперечливі оцінки слабких класифікаторів у відсотках для кожного класу виразів обличчя. Наприклад, на малюнку видно, що 42,5% векторів ознак правильно класифікують злісні міфічні вирази, тоді як 18,7% векторних ознак класифікують їх як огиду, 9,4% елемента вектора класифікують їх як страшні, 7,6%.

Тому ми можемо бачити, що в основному гнівні вирази обличчя плутаються з огидою і сумними; огидна обличчя висловлюються з розумом; страшні вирази обличчя плутаються з щасливими та здивованими, щасливі вирази обличчя плутають з страшними та огидними; сумні обличчя висловлюються з розумом; і здивований вираз обличчя плутають з страшними.

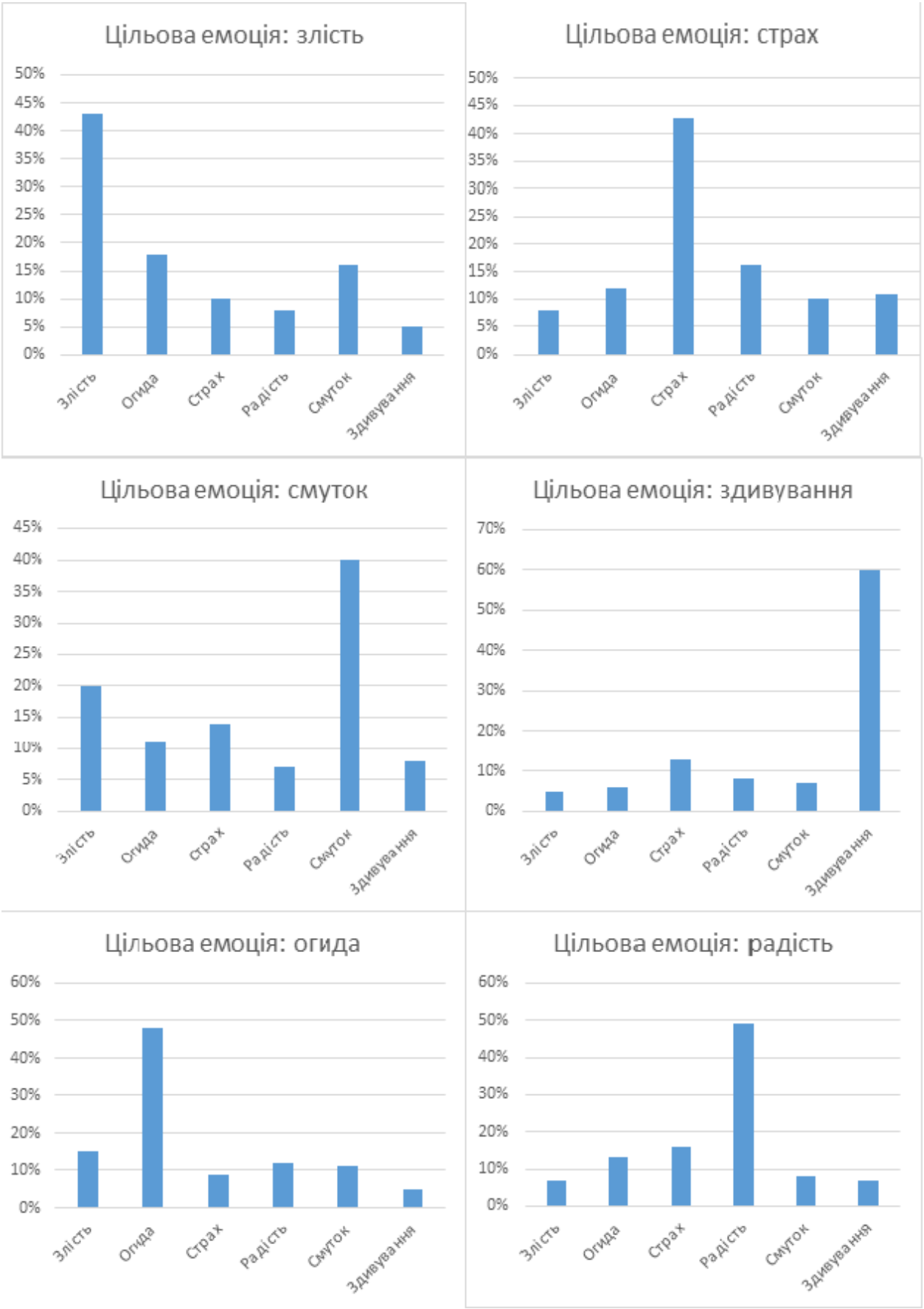


Рисунок 2.4 – Суперечливі складові при розпізнавання шести базових емоцій

## 2.8 Розпізнавання емоцій на основі методу опорних векторів SVM

SVM - це клас алгоритму лінійної класифікації, який має на меті знайти відокремлюючу гіперплощину з максимально можливою різницею між двома різними категоріями даних. У нашому експерименті ми використовуємо публічно доступну реалізацію SVM, в якій ми використовували ядро радіальної базової функції (RBF), і оптимальний вибір параметрів здійснюється на основі стратегії пошуку сітки.

Як вказано раніше, це два типи функцій від результатів відстеження значущості, що використовуються в нашій системі. Коли AdaBoost виділяє вектори функцій, для класифікації виразів обличчя, що використовують SVM, ми генеруємо новий набір функцій із набору векторів функцій, вибраних за допомогою AdaBoost. Для того, щоб зберегти розмір функції, що використовується в SVM класифікації якнайменше, ми взяли максимальне значення переміщення у двох напрямках від вектора функції, визначеного в рівнянні (2.6), і максимального зміни кута і відстані від вектора функції, визначений у рівнянні (2.9). Вектором властивостей, визначеним у рівнянні (2.6), який пов'язаний з результатом відстеження і-го орієнтира в k-тій послідовності виразів, отримуємо наступні два значення:

$$\delta x'^{\mathrm{k}}_{i\,max} = max\{\delta x'^{\mathrm{k}}_{1,i}, \delta x'^{\mathrm{k}}_{2,i}, \dots, \delta x'^{\mathrm{k}}_{N,i}\}$$
$$\delta y'^{\mathrm{k}}_{i\,max} = max\{\delta y'^{\mathrm{k}}_{1,i}, \delta y'^{\mathrm{k}}_{2,i}, \dots, \delta y'^{\mathrm{k}}_{N,i}\}$$

Аналогічним чином, вектор ознак, визначений у рівнянні (2.9), який пов'язаний з результатами відслідковування і-го елемента когерентної послідовності виразів, надає наступні два значення:

$$\delta\theta'^{\mathrm{k}}_{(i,j)max} = max\{\delta \mathrm{d}'^{\mathrm{k}}_{1(i,j)}, \delta \mathrm{d}'^{\mathrm{k}}_{2(i,j)}, \dots, \delta \mathrm{d}'^{\mathrm{k}}_{N(i,j)}\}$$
$$\delta\theta'^{\mathrm{k}}_{(i,j)max} = max\{\delta\theta'^{\mathrm{k}}_{1(i,j)}, \delta\theta'^{\mathrm{k}}_{2(i,j)}, \dots, \delta\theta'^{\mathrm{k}}_{N(i,j)}\}$$

Вихідний процес цього вектора є єдиним вектором для кожного відео. Розміри функції залежать від кількості функцій, вибраних AdaBoost. Якщо використовувати

вектори L, вибрані AdaBoost, розмірність функції для класифікації SVM буде L × 2. Експерименти показують, що більше ніж 90% функцій були вибрані з функцій другого типу, тобто вектори функцій, створені парами орієнтирів в результаті відстеження. Це доводить, що рухи орієнтирів, як виражають особливий вираз обличчя, не є незалежними.

Таблиці 2.5, 2.6 та 2.7 показують матриці складові кожної з емоцій для розпізнавання виразу обличчя за допомогою вибраних функцій 100, 200 та 400 AdaBoost з розмірністю 200, 400 та 800 відповідно. Середня точність розпізнавання становила 93,2%, 95,5% та 97,3%, використовуючи відповідно 200, 400 та 800 розмірів. Отримуємо покращення в 2,1% для розпізнавання виразів обличчя за допомогою SVM з покращеними функціями.

Таблиця 2.5 – Точність розпізнавання емоцій з використанням методу опорних векторів та покращеного класифікатора (100 вибраних ознак класифікатора)

| % | Злість | Огида | Страх | Щастя | Сум | Здивування |
|---|---|---|---|---|---|---|
| Злість | 92.5 | 2.5 | 0 | 0 | 5 | 0 |
| Огида | 0 | 96.6 | 1.6 | 0 | 1.6 | 0 |
| Страх | 0 | 0 | 92 | 8 | 0 | 0 |
| Щастя | 0 | 3.1 | 1.5 | 95.3 | 0 | 0 |
| Сум | 6.6 | 0 | 3.3 | 0 | 87.6 | 3.3 |
| Здивування | 0 | 0 | 2.5 | 0 | 2.5 | 95 |

Таблиця 2.6 – Точність розпізнавання емоцій з використанням методу опорних векторів та покращеного класифікатора (200 вибраних ознак класифікатора)

| % | Злість | Огида | Страх | Щастя | Сум | Здивування |
|---|---|---|---|---|---|---|
| Злість | 93 | 2 | 5 | 0 | 0 | 0 |
| Огида | 1.6 | 96.6 | 1.67 | 0 | 0 | 0 |
| Страх | 4 | 0 | 84 | 8 | 0 | 4 |
| Щастя | 0 | 3.1 | 0 | 96.9 | 0 | 0 |
| Сум | 3.3 | 0 | 0 | 0 | 96.6 | 0 |
| Здивування | 0 | 0 | 0 | 0 | 1.2 | 98.7 |

Таблиця 2.7 – Точність розпізнавання емоцій з використанням методу опорних векторів та покращеного класифікатора (400 вибраних ознак класифікатора)

| % | Злість | Огида | Страх | Щастя | Сум | Здивування |
|---|---|---|---|---|---|---|
| Злість | 100 | 0 | 0 | 0 | 0 | 0 |
| Огида | 1.6 | 96.6 | 1.6 | 0 | 0 | 0 |

Продовження таблиці 2.7

| | | | | | | |
|---|---|---|---|---|---|---|
| Страх | 0 | 0 | 92 | 4 | 0 | 4 |
| Щастя | 0 | 0 | 0 | 100 | 0 | 0 |
| Сум | 0 | 0 | 3.33 | 0 | 96.6 | 0 |
| Здивування | 0 | 0 | 0 | 0 | 1.25 | 98.7 |

## 2.9 Порівняння результатів роботи алгоритмів

На основі запропонованої методики досягнута, точність розпізнавання на наборі даних Cohn-Kanade для розпізнавання виразів обличчя, результати порівнянно з сучаснми алгоритмами розпізнавання. Усі дослідження проводилися за допомогою безкоштовної програми, ознайомитись з можливостями та функціями якої можна за посиланням [35]. Досягнуто 93,1% точності розпізнавання обличчя, використовуючи багатокласові алгоритми AdaBoost з подібністю DTW між слабким класифікатором на основі векторних ознак і 95,3% точності розпізнавання за допомогою SVM на підвищених функціях. Система показала чудові показники і досягла швидкості розпізнавання 97,7%. У даному методі ініціалізація орієнтира є ручним процесом, а кількість орієнтирів також більше, ніж кількість орієнтирів у запропонованому методі. З іншого боку, запропонований метод є повністю автоматичним. Було досягнуто 94,2% точності розпізнавання шести базових емоцій, використовуючи метод, заснований на багатокласових AdaBoost шаблонах і SVM-класифікаторах.

Основний недолік методу полягає в тому, що він перевіряється лише в ідеальних послідовностях зображень, вирівняних вручну, і експериментів у повністю автоматичних умовах не було представлено.

Інший, більш пізній спосіб, запропонований Чжаном та ін. [36] досяг 97,14% точності розпізнавання. Функції LBP використовувалися з розрідженим класифікатором подання. Тому найкраща точність розпізнавання різних методів, запропонована дослідниками в літературі, становить близько 97% (за винятком методу в ), на базі даних Cohn-Kanade виразів обличчя. Наш запропонований спосіб також досягав більш ніж 94% точності розпізнавання, що є другою найкращою точністю до цих пір, принаймні, за даними авторів.

## 2.10 Висновки до другого розділу

Результати дослідження показують, що рухи орієнтирів на обличчі, як певний вираз, еволюціонують та не залежать один від одного. У запропонованій системі прототипні вирази обличчя обчислюються на основі припущення, що кожен вираз обличчя можна змоделювати за допомогою унімодільних розподілів. Це означає, що для кожного класу виразів обличчя в базі даних існують схожі переміщення орієнтирів, оскільки вираз обличчя розвивається з часом, незалежно від етнічної групи, віку та статі.

Запропоновано розширення алгоритму AdaBoost і порівняно з сучасним алгоритмом виявлення обличчя Віоли та Джонса. Запропонований алгоритм TCA знаходить останній класифікатор шляхом мінімізації верхньої межі помилки тренувань і створює значно меншу класифікацію. Отримані результати в порівнянні з оригінальним алгоритмом AdaBoost з точки зору виявлення обличчя та розпізнавання емоцій кращі приблизно на 2,6%. Класифікатор, навчений за новим методом, був приблизно на 4% швидше, до того ж він складається лише з чверті слабких класифікаторів, необхідних для класифікатора, навченого стандартним AdaBoost. Алгоритм може бути застосований з іншими слабкими класифікаторами, придатними для виявлення обличчя.

# 3 МОДЕЛЮВАННЯ І ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА

## 3.1 Відстеження ключових точок на обличчі людини

Розпізнавання обличчя в основному складається з трьох підсистем: відстеження ключових точок обличчя, створення ознаки на основі результату відстеження цих точок та класифікації отриманої ознаки.

Алгоритм починається, створюючи графік вузлів обличчя. Кожен вузол графічного з'єднання відповідає обличчю еталону і містить декілька фільтрів Габора, які отримані із зображення моделі. Габари - це сукупність складних коефіцієнтів Габора з того ж місця на зображенні. Коефіцієнти утворюються за допомогою вейвлетів Габора різноманітних розмірів, орієнтацій та частот. Графік вузол служить базою даних дескрипторів опису, які можна використовувати для розміщення орієнтирів у нових зображеннях.

На рисунку 3.1 показано узагальнений процес ініціалізації та відстеження ключових точок. Знаходження цих точок в новому образі має два етапи. По-перше, місце розташування орієнтирів оцінюється на основі відомих розташувань інших орієнтирів на зображенні, по-друге, ця оцінка покращується шляхом витягання потоку моделей Gabor з цього зображення на приблизних місцях і його порівняння із наперед заданими прототипними моделями.

Для того, щоб зробити систему повністю автоматичною, спочатку потрібні приблизні місця розташування принаймні одного або двох орієнтирів. Ця мета досягається спочатку локалізацією області обличчя на зображенні за допомогою методу виявлення обличчя на основі ознак Хаара. Тепер орієнтири в центрі двох очей шукаються в області обличчя, використовуючи той самий Haar-подібний метод визначення об'єктів на основі функцій, запропонований Віолою та Джонсом [36].

Пошук розташування інших орієнтирів виконується досить легко, на основі відомих координат очей. Кожне нове визначне місце розташування оцінюється на основі попередньо локалізованих елементів обличчя. Значне місце розташування потім уточнюється, порівнюючи струмінь Габор, що витягнутий із розрахункової точки, до відповідної моделі з графіка групи. Процес повторюється до тих пір, поки не знайдуться

всі місця. Ці місця це 52 ініціалізовані орієнтири у зображенні нейтрального обличчя (перший кадр відеозапису).

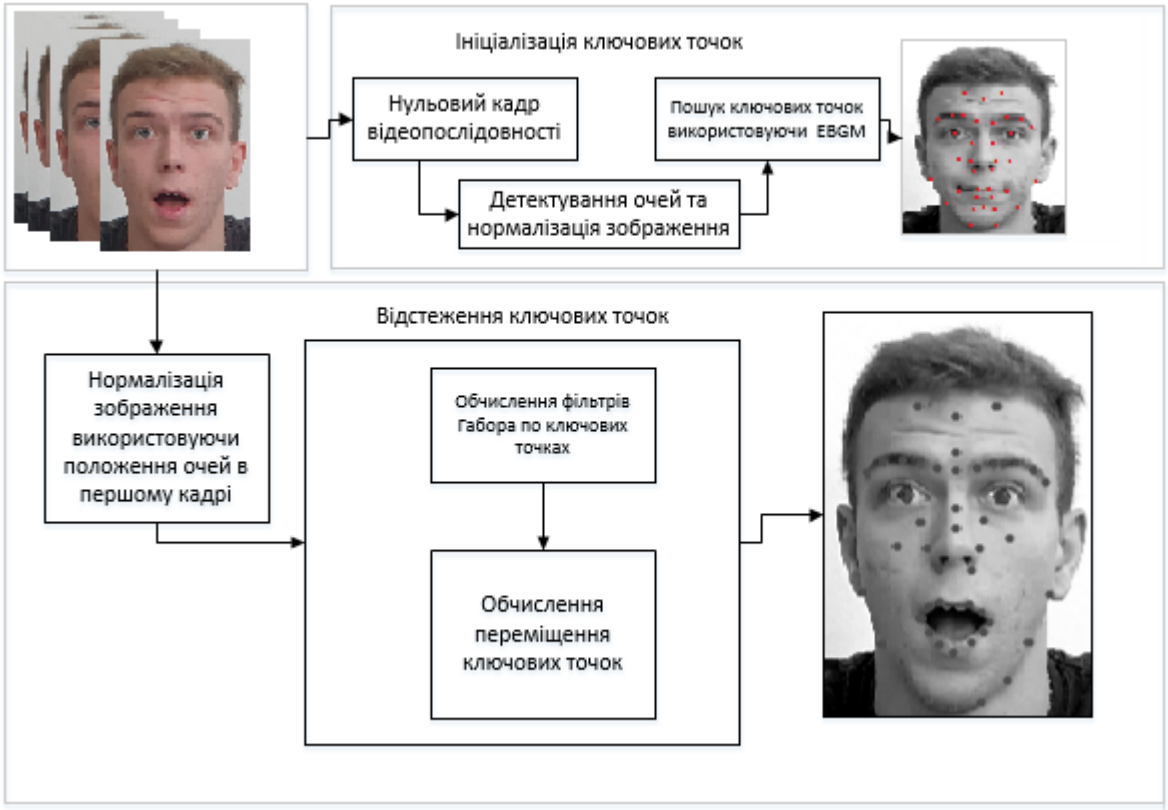


Рисунок 3.1 – Ініціалізація ключових точок на зображенні

Орієнтири ініціалізуються в першому кадрі відеопослідовності, наступним кроком буде відстеження, оскільки в кожен окремий момент часу орієнтири змінюють своє положення відносно очей на обличчі. У кожному вхідному кадрі слід оцінити зміщення орієнтирів відносно попереднього кадру. Зсув орієнтирів по відношенню до попереднього кадру може бути безпосередньо розрахований з використанням рівняння для оцінки прямого зсуву. Це переміщення дає точне положення ключових точок в поточному кадрі. Для кожної точки в поточному кадрі відбувається оновленння, і цей самий процес повторяється, щоб знайти переміщення орієнтирів у наступному кадрі. Багатообіцяючий результат відстеження орієнтирів отримується за допомогою цієї концепції. На рисунку 3.2 показано результат відстеження орієнтирів у деяких послідовностях виразів обличчя; на рисунку показані лише декілька зображень з кожної послідовності.

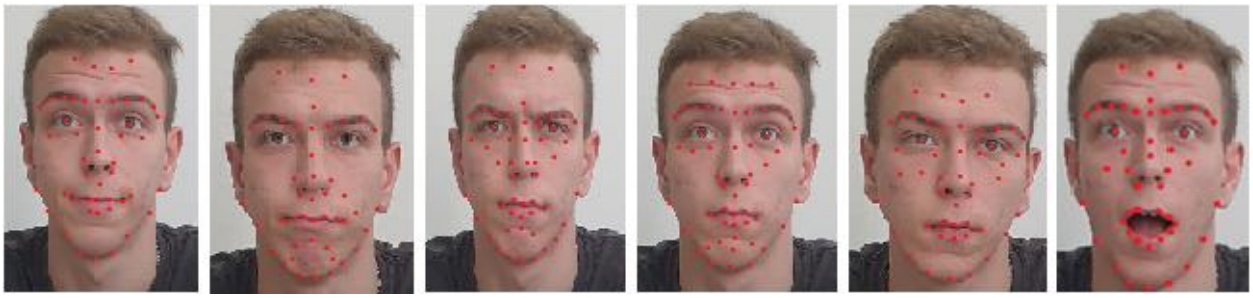

Рисунок 3.2 – Відстеження ключових точок

## 3.2. Нормалізація ключових точок

Нормалізація орієнтира відносить кожен орієнтир до рівномірної координатної позиції у першому кадрі відеозйомки, і, коли вираз еволюціонує у часі, орієнтири відповідно зміщуються. Нехай $S_i^k$ є результатом відстеження і-го орієнтира в k-й послідовності виразу:

$$S_i^k = \{(x_0, y_0)_i^k, (x_1, y_1)_i^k, (x_2, y_2)_i^k, \dots (x_{N-1}, y_{N-1})_i^k, (x_N y_N)_i^k\}, \quad (3.1)$$

де $(x_1, y_1)_i^k$ є і-тою орієнтирною позицією координати в 1-ому кадрі послідовності k-ої експресії, а N - число кадрів у послідовності розпізнавання.

Середня позиція ключової точки, яка відповідає кожному орієнтиру, обчислюється з усіх нейтральних зображень, що є першим кадром у кожному знімку. Припустимо, що $(\mu_{x0}, \mu_{y0})$ позначає середнє значення ключової точки позиції і-го орієнтиру в першому кадрі послідовностей виразів обличчя. Для кожного результату відстеження, для якого орієнтири мають бути нормалізовані, визначається різниця між першим орієнтиром та середнім орієнтиром. Це дає зміщення орієнтира у відношенні до середньої знакової позиції. Позначимо $(\delta_{x0}\delta_{x0})_i^k$ зміщення і-го орієнтира в першому кадрі k-ої послідовності вираження емоції відносно середньої орієнтирної позиції:

$$(\delta_{x0}\delta_{x0})_i^k (\mu_{x0} - x_0, \mu_{y0} - y_0)_i^k \quad (3.2)$$

Переміщення, що відповідає кожній ключовій точці додано до орієнтирів позицій у кожному кадрі послідовності виразів обличчя. Трансформований результат орієнтирного відстеження тепер позначається $S_i'^{\mathrm{k}}$ і визначається як:

$$S_i'^{\mathrm{k}} = \{(x_0 + \delta_{x0}, y_0 + \delta_{y0})_i^k, (x_1 + \delta_{x0}, y_1 + \delta_{y0})_i^k, \dots, (x_N + \delta_{x0}, y_N + \delta_{y0})_i^k\} \quad (3.3)$$

Результат відстеження нормалізується таким чином, що кожен орієнтир для всієї послідовності виразу починається з тієї ж позиції координат, тобто $(\mu_{x0}, \mu_{y0})$ і розвивається відповідно до зміщення в наступних кадрах. На рисунку 3.3 показано результати відстеження орієнтирів перед (перший рядок) та після (другий рядок), нормалізацією (зверніть увагу, що краї між орієнтирами наведені лише для того, щоб зробити обличчя).

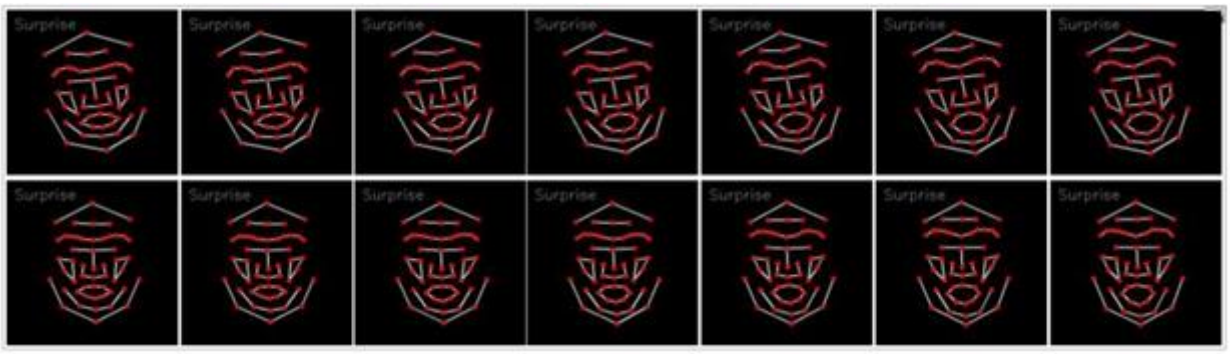

Рисунок 3.3 – Результати відстеження ключових точок

## 3.3 Вибір ознак для класифікації

У запропонованому підході розпізнавання емоції на обличчі виконується виключно за геометричною інформацією, без безпосереднього взяття будь-якої інформації про текстуру обличчя. Набір ознак складається з двох типів ознак, одним з яких є можливість розгляду результатів відстеження кожної окремої ключової точки обличчя, а інший - шляхом розгляду результатів відстеження пар ключових точок. Нехай (x', y') - це позиція координат, що відповідає певній ключовій точці та змінюється в часі, тоді перерахуємо рівняння (3.3) у такому вигляді:

$$S_i'^{\mathrm{k}} = \left\{(x'_0 y'_0)_i^k, (x'_1 y'_1)_i^k, \ldots, (x'_N y'_N)_i^k\right\} \tag{3.4}$$

Кількість кадрів у різних відеороликах виразу обличчя може бути різною. Як буде описано далі, підмножина вибору функції з наявного пулу функцій виконується за допомогою AdaBoost, з подібністю DTW між прототипним вектором та вектором функції введення як слабким класифікатором. Тому для кожного класу виразів обличчя слід створити прототипний вираз обличчя. Для цього кількість кадрів повинна бути рівною у

кожній послідовності виразів обличчя в базі даних. Після завершення створення прототипу та вибору функцій кількість кадрів у послідовності не обов'язково має бути рівною на етапі класифікації.

Розглянемо у рівнянні (3.4), що кожен вираз емоції на обличчі має однакове число кадрів. На практиці ми змінюємо розмір послідовності відстеження віхи, використовуючи лінійну інтерполяцію. У нашому експерименті ми використовуємо N = 15, а також L = 52 орієнтирів обличчя. Функціональний тип - це вектор функції від окремої послідовності відстеження віхи. Кожна орієнтирна координата в послідовності тепер віднімається від першої координатного орієнтиру, тобто позиції орієнтації в нейтральному кадрі, для створення вектору ознак типу 1. Припустимо, що $(\delta x_1' \delta y_1')_i^k$ означає різницю в і-му орієнтирі в 1-м кадрі, починаючи з і-го орієнтира в першому кадрі k-го відеозапису:

$$\left(\delta_{x_l'} \delta_{y_l'}\right)_i^k = (x_l' - x_0', y_l' - y_0')_i^k \tag{3.5}$$

Обчислюємо вектор ознак в першому кадрі:

$$\delta S_i'^{\mathrm{k}} = \left\{(\delta x_1' \delta y_1')_i^k, (\delta x_2' \delta y_2')_i^k, \dots, (\delta x_N' \delta y_N')_i^k\right\} \tag{3.6}$$

Далі вектор функції створюється з пари орієнтирів у послідовності вираження. Ми називаємо цю функцію вектором функції типу два. По-перше, розраховується кутовий та евклідовий відстань між кожною парою орієнтирів усередині рами. Припустимо $, (d_0', \theta_0')_{i,j}^k$ позначає відстань та кут між і-тою та j-тою парою орієнтирів у 1-му кадрі послідовності k-ої експресії. Позначимо розрахункову послідовність відстані та кута як $G_{i,j}'^{\mathrm{k}}$.

$$G_{i,j}'^{\mathrm{k}} = \left\{(d_0', \theta_0')_{i,j}^k, (d_1', \theta_1')_{i,j}^k, \dots, (d_N', \theta_N')_{i,j}^k\right\} \tag{3.7}$$

Тепер ці відстані та кути віднімаються від відповідної відстані та кутів на першому кадрі відеозапису. Припустимо, що $(\delta d_l', \delta\theta_l')_{i,j}^k$ позначає зміну відстані та кута між і-ї та j-ої парними орієнтирами в 1-шому кадрі по відношенню до першого кадру в 1-му відеозйомці:

$$(\delta d_l', \delta\theta_l')_{i,j}^k = (d_l' - d_0', \theta_l' - \theta_0')_{i,j}^k \tag{3.8}$$

Обчислюємо вектор ознак в другому кадрі відеопослідовності:

$$\delta G_{i,j}^{\prime k} = \left\{(\delta d_0', \delta\theta_0')_{i,j}^k, (\delta d_1', \delta\theta_1')_{i,j}^k, \dots, (\delta d_N', \delta\theta_N')_{i,j}^k\right\} \quad (3.9)$$

Функція типу 1 має вектори функцій L = 52, а функція типу two має M = L × (L-1) / 2 = 1,326 векторних ознак. Загалом, в базі функцій є L + M = 1,378 векторних ознак. З тих векторних ознак лише деякі з них містять більшу частину дискримінаційної інформації для розпізнавання міміки. Схема вибору функцій AdaBoost використовується для вибору підмножини векторних функцій з пулу властивостей, достатнього для розпізнавання мімічних виразів.

## 3.4 Збір даних для кодування емоцій

Для пошуку векторного простору характеристичних ознак, побудови базису цього простору, відтворення похідних емоційних станів, виконано наступне:

– створена множина фотографічних зображень, на якій актором відтворювались ситуації ξ1, ξ2, ξ3 в яких виникають базові емоції, та описана міміка, притаманна цим емоціям;

– проаналізована отримана множина з метою виявлення областей, які містять характеристичні ознаки емоцій та їх опис (використовуючи анатомічні ознаки та методику FACS);

– створено у просторі характеристичних ознак базис для наступного розкладу по ньому довільних векторів мімічних проявів емоційних станів.

У табл. 3.1 наведено результати досліджень – описи ситуацій, у яких виникають базові емоції, відповідні їм фотографічні зображення та опис міміки, яка характеризує ці стани.

У ході дослідження отримано 21 характеристичну ознаку, комбінація яких утворює базис мімічних проявів емоцій (табл. 3.2).

Мімічні вирази емоції (*Em*) були подані у вигляді вектора:

$$Em_i^n = (\mu_1, \mu_2, \dots, \mu_{21}),\ i = \overline{1,6}, \quad (3.10)$$

де $\mu_m \in [0; 1]$ – характеристична мімічна ознака (при μ = 0 – немає ознаки, а при μ = 1 вплив ознаки максимальний).

Таблиця 3.1 – Фотоеталони базових емоцій

| Базова емоція | Ситуація в якій виникає емоція (ξ1, ξ2, ξ3) | Зображення емоції | Опис міміки в розрізі зон обличчя (1 - область чола й брів; 2 - область очей; 3 - нижня частина обличчя) |
|---|---|---|---|
| Сум | ξ1=0 – неможливість задоволення потреби;<br>ξ2=1 – виникає після того як відбулася неприємна подія;<br>ξ3=1 – є переживанням втрати. |  | 1. Внутрішні кутики брів підняті вверх;<br>2. Внутрішні кутики верхніх повік припідняті;<br>3. Рот закритий, кутики губ опущені. |
| Радість | ξ1=1 – потреба досягнута;<br>ξ2=1 – виникає після задоволення потреби;<br>ξ3=1 – є переживання результату задоволення потреби. |  | 1. Брови та чоло спокійні;<br>2. У зовнішнього краю кутиків очей зморшки - «гусячі лапки»;<br>3. Кутики губів відтягнуті в сторони та припідняті. Від носа до зовнішнього краю губів зморшки. |
| Страх | ξ1=0 – передчуття незадоволення потреби;<br>ξ2=0 – передчуття втрати;<br>ξ3=1 – боязнь за себе і свої потреби. |  | 1. Брови підняті та зведені. Зморшки в центрі чола;<br>2. Верхні повіки підняті так, що видно склеру, а нижні припідняті та напружені;<br>3. Рот розкритий, губи розтягнуті та напружені. |
| Гнів | ξ1=0 – негативна емоція;<br>ξ2=1 – виникає після події яка привела до незадоволення потреби;<br>ξ3=0 – направлений на об’єкт, який заважає досягненню мети. |  | 1. Брови опущені і зведені, між бровами вертикальні зморшки;<br>2. Верхні повіки напружені, нижні повіки напружені і припідняті;<br>3. Рот закритий, губи затиснуті. |
| Здивування | ξ1=1 – задоволення потреби;<br>ξ2=0 – передчуття задоволення потреби;<br>ξ3=0 – направлений на об’єкт. |  | 1. Брови припідняті, на лобі зморшка;<br>2. Повіки трохи розширені;<br>3. – . |
| Огида | ξ1=0 – невдоволення потреби;<br>ξ2=0 – передчуттям незадоволення потреби;<br>ξ3=0 – направлена на об’єкт. |  | 1. Брови припідняті;<br>2. –;<br>3. Кутики губ опущенні. Обличчя зморщене, голова піднесена, наче людина відсторонюється від співрозмовника. |

Таблиця 3.2 – Мімічні прояви для формування базових емоцій

| ознака | Опис мімічних проявів у розрізі областей обличчя | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Область обличчя | | Мімічний прояв | Радість | Горе | Страх | Гнів | Здивування | Зневага |
| μ1 | Область чола і брів | Чоло | Зморшки в центрі чола | 0 | 0 | 1 | 0 | 0 | 0 |
| μ2 | | | Одна горизонтальна зморшка | 0 | 0 | 0 | 0 | 1 | 0 |
| μ3 | | | Між бровами горизонтальна зморшка | 0 | 0 | 0 | 1 | 0 | 0 |
| μ4 | | Брови | Внутрішні кутики підняті вверх | 0 | 1 | 0 | 0 | 0 | 0 |
| μ5 | | | Опущені та зведені | 0 | 0 | 0 | 1 | 0 | 0 |
| μ6 | | | Припідняті | 0 | 0 | 0 | 0 | 1 | 1 |
| μ7 | | | Підняті та зведені | 0 | 0 | 1 | 0 | 0 | 0 |
| μ8 | Область очей (очі, повіки, основа носа) | Верхні повіки | Внутрішні кутики підняті | 0 | 1 | 0 | 0 | 0 | 0 |
| μ9 | | | Напружені | 0 | 0 | 0 | 1 | 0 | 0 |
| μ10 | | | Підняті (видно склеру) | 0 | 0 | 1 | 0 | 0 | 0 |
| μ11 | | | Припідняті | 0 | 0 | 0 | 0 | 1 | 0 |
| μ12 | | Нижні повіки | Припідняті та ненапружені | 1 | 0 | 0 | 0 | 1 | 0 |
| μ13 | | | Припідняті та напружені | 0 | 0 | 1 | 1 | 0 | 0 |
| μ14 | | Зморшки | "Гусячі лапки" біля зовнішніх кутиків | 1 | 0 | 0 | 0 | 0 | 0 |
| μ15 | | | Зморшка під повіками | 1 | 0 | 0 | 0 | 0 | 0 |
| μ16 | Нижня частина обличчя (ніс, щоки, рот) | Рот | Закритий, губи стиснуті | 0 | 0 | 0 | 1 | 0 | 0 |
| μ17 | | | Розкритий | 0 | 0 | 1 | 0 | 0 | 0 |
| μ18 | | Губи (лінія, кутики) | Кутики губ відтягнуті в сторони та припідняті | 1 | 0 | 0 | 0 | 0 | 0 |
| μ19 | | | Розтягнуті та напружені | 0 | 0 | 1 | 0 | 0 | 0 |
| μ20 | | | Кутики губ опущені | 0 | 1 | 0 | 0 | 0 | 1 |
| μ21 | | Зморшки | Зморшка від носа до кутиків губ | 1 | 0 | 0 | 0 | 0 | 0 |

Отриманий набір шести векторів утворює базис $B_{i,j}, i = \overline{1,21}$, j = $\overline{1,6}$ векторного простору мімічних проявів емоційних станів.

У запропонованій формальній моделі базис простору мімічних ознак емоційних станів будується на основі апріорного досвіду експериментатора, вимагає певної кваліфікації і, відповідно, дає неоднозначний результат – констатація одного і того ж

емоційного м’язового прояву в різних людей розрізняється. Для того, щоб перейти від феноменологічного визначення характеристичних мімічних ознак до певної їх формалізації, запропоновано використати власну модифікацію методу моделей, які деформуються. В якості параметричних кривих, для виділення характерних ознак рис обличчя, запропоновано моделі які задаються з допомогою нерівномірних раціональних базисних сплайннів.

## 3.5 Аналіз результатів дослідження алгоритму розпізнавання емоцій

Виходячи з аналізу проведеної роботи, а саме дослідження алгоритму для розпізнавання емоцій на обличчі людини Adaboost з використанням ознак Хаара та реалізації відповідної системи можна в першу чергу встановити відповідність між емоціями та регіонами на обличчі які відповідають за розпізнавання конкретної емоції (табл. 3.3).

Таблиця 3.3 – Регіони обличчя що є ключовими і максимально значущими при розпізнавання конкретних емоцій.

| Базова емоція | Елементи обличчя |
|---|---|
| Здивування | Брови, горизонтальні зморшки на лобі, повіки, нижня щелепа, губи, зуби |
| Страх | Брови, зморшки в центральній частині лоба, повіки, рот, губи |
| Відраза | Губи, зморшки на носі, щоки, зморшки на шкірі під нижніми повіками, повіки, брови |
| Гнів | Брови, вертикальні зморшки між бровами, повіки, губи, ніздрі |
| Радість | Кутики рота, рот, губні складки, щоки, повіки, зморшки під повіками, зморшки у вигляді ”гусячих лапок від зовнішніх кутиків очей до скронь” |
| Сум | Внутрішні кутики брів, внутрішні кутики верхніх брів, кутики рота, губи |

Отже, для розпізнавання виразів обличчя потрібно розглядати такі елементи обличчя:

- зовнішні та внутрішні кінці брів, а також їх середина;
- місця з’єднання верхніх та нижніх повік;
- середина верхніх та нижніх повік;

- кутики губ;
- середина верхньої та нижньої губи;
- підборіддя;
- зіниці;
- кінчик носа;
- наявність зморщок на чолі;
- наявність зморщок між бровами;
- наявність зморщок на носі;
- наявність зморщок на підборідді.

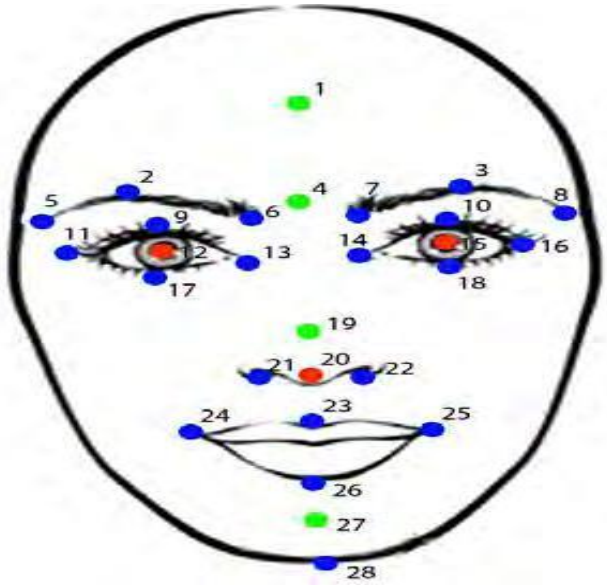


Рисунок 3.4 – Елементи які відіграють важливу роль у визначенні емоції

На рисунку 3.4 зображено регіони обличчя під номером 1,4,19,27 – елементи, що відповідають присутності чи відсутності зморщок; Елементи що співпадають з областю центру очей – базові точки, які відідграють важливу роль для відстежуваня області обличчя. Відносно цих точок відбувається нанесення на зображення масок для пошуку таких частин обличчя як брови, ніс, губи, та лоб. Всі інші точки – елементи, місцезнаходження яких важливе для виявлення емоції

## 3.6 Аналіз роботи реалізованої системи

### 3.6.1 Тестування правильності розпізнавання емоцій

Тестування проводилось на основі шести базових емоцій, такі як радість, сум, злість, здивування, гнів, страх. Актор виражав вище вказані емоції (рис. 3.2) у випадковій послідовності.

Для досягнення результату тестування, актором було виражено 60 емоцій для кожного виду емоції по 10 виразів обличчя (табл. 3.4).

Таблиця 3.4 – Тестування розпізнавання емоцій

| Номер спроби | Емоція що виражадась | Розпізнана емоція | Помилки, «No face» |
|---|---|---|---|
| 1 | радість | радість | – |
| 2 | радість | радість | – |
| 3 | радість | радість | – |
| 4 | сум | сум | – |
| 5 | здивування | страх | хибне розпізнавання |
| 6 | огида | огида | – |
| 7 | злість | злість | – |
| 8 | сум | сум | – |
| 9 | страх | страх | – |
| 10 | огида | огида | – |
| 11 | страх | страх | – |
| 12 | сум | сум | – |
| 13 | сум | сум | – |
| 14 | здивування | здивування | – |
| 15 | радість | радість | – |
| 16 | радість | радість | – |
| 17 | злість | злість | – |
| 18 | страх | страх | – |
| 19 | злість | злість | – |
| 20 | здивування | здивування | – |
| 21 | здивування | здивування | – |
| 22 | страх | страх | – |
| 23 | страх | страх | – |
| 24 | сум | сум | – |
| 25 | сум | огида | хибне розпізнавання |
| 26 | сум | сум | – |
| 27 | злість | злість | – |
| 28 | огида | огида | – |
| 29 | здивування | здивування | – |

Продовження таблиці 4.2 – Тестування розпізнавання емоцій

| Номер спроби | Емоція що виражалась | Розпізнана емоція | Помилки «No face» |
|---|---|---|---|
| 30 | страх | страх | – |
| 31 | злість | злість | – |
| 32 | здивування | здивування | – |
| 33 | огида | – | no face |
| 34 | радість | радість | – |
| 35 | радість | радість | – |
| 36 | огида | огида | – |
| 37 | радість | радість | – |
| 38 | радість | радість | – |
| 39 | огида | огида | – |
| 40 | злість | злість | – |
| 41 | огида | огида | – |
| 42 | здивування | здивування | |
| 43 | злість | злість | |
| 44 | страх | здивування | хибне розпізнавання |
| 45 | сум | сум | – |
| 46 | сум | огида | – |
| 47 | здивування | здивування | |
| 48 | огида | огида | – |
| 49 | злість | злість | – |
| 50 | злість | злість | – |
| 51 | радість | радість | – |
| 52 | страх | страх | – |
| 53 | огида | огида | – |
| 54 | здивування | здивування | – |
| 55 | здивування | здивування | – |
| 56 | злість | злість | – |
| 57 | сум | сум | – |
| 58 | огида | огида | – |
| 59 | здивування | здивування | – |
| 60 | страх | здивування | хибне розпізнавання |

Аналізуючи результати проведеного тестування, можна зробити висновки що реалізована майже ідеально справляється з завданням визначення області обличчя що у нашому випадку є першим етапом на шляху до розпізнавання емоцій, так як без її визначення неможлива подальша коректна робота програми. І становить цей показник близько 98,4% від загальної кількості спроб.

Необхідно зазначити що не всі емоції розпізнаються однаково добре. Для прикладу можна сказати що найкраще розпізнається така емоція як радість. Під час всього експерименту не було ні одного хибного розпізнавання, а також вказана емоція жодного разу не була сплутана з іншої емоцією.

Що стосується таких емоцій як сум, огида, здивування, злість, то можна сказати що вони відносяться до середнього рівня складності розпізнавання. Так як вони або були хибно розпізнані або ж іншу емоцію приймали за одну з них.

Можна назвати одну з найважчих емоцій для розпізнавання, це страх. Виходячи з того що при розпізнаванні даної емоції траплялося найбільше помилок. Двічі вказану емоцію система розпізнавала як здивування, ще одна ситуація була коли здивування було розпізнано як страх. Можливо така ситуація пов’язана зі складністю вираження емоцій для людини в ситуаціях в яких зазвичай неприсутні вказані емоції.

### 3.6.2 Тестування роботи програми за різних умов

Робота система тестувалась в різних умовах, з різними кутами нахилу та повороту голови актора. Також в ситуаціях коли деякі елементи обличчя закриті, або ж частина обличчя взагалі не потрапляє на первинний датчик у нашому випадку вмонтованої веб-камери, за допомогою якої відбувається розпізнавання.

Коли поворот голови актора був в діапазоні від 10 до 20 градусів у ліву чи праву сторону працездатність системи як і точність розпізнавання залишалась на високому рівні та майже не змінювалась. Проте необхідно зазначити що, якщо кут повороту був більший ніж 15 градусів, в системі явно зменшується кількісне сприйняття емоції.

Що стосується нахилу голови по вертикалі вниз або вверх, то можна спостерігати гірші результати в порівнянні випадком коли був поворот по горизонталі. Видно що погіршується правильність визначення та окреслення області обличчя. Також починається погіршуватися сприйняття програмою сили з якою виражена емоція, особливо при нахилі голови вниз. Значення правильності розпізнавання емоції вже є граничним, між правильно розпізнаної емоцією та хибним розпізнаванням.

Важливо відзначити що при нахилі голови в сторону на кут більше ніж 45 градусів система перестає як локалізовувати область обличчя так і розпізнавати емоції. Що є суттєвим недостатком. Така поведінка зумовлена тим, що для підвищення точності та швидкості розпізнавання накладання масок для визначеня області обличчя та розпізнавання не відбувається при нахилі головни в праву ли ліву сторону на кути більш

ніж 15 градусів. Системі також не вдається окреслити обличчя при повному повороті голови в праву чи ліву

## 3.7 Висновки до третього розділу

Описано алгоритм пошуку та нормалізації ключових точок на обличчі людини. Проведено збір даних для кодування емоційних виразів, встановлено відповідність між регіонами обличчя та емоціями за вираження яких вони відповідають. Розроблено функціональні та нефункцірнальні вимоги до майбутньої системи.

На основі проведених дослідженнь та запропонованої методики розпізнавання емоцій реалізовано інформаційну модель. Для реалізації моделі використовувалась мова програмування python та середовище розробки Pycharm. Також в процесі реалізації використовувались набір бібліотек комп’ютерного зору одна із яких OpenCV. В процесі реалізації інформаційної системи були використані сучасні засоби та технології для розробки, що дозволило повною мірою реалізувати функції ІС.

Використання фреймворків для створення інтерфейсу інформаційної системи стало результатом того, що отриманий користувацький інтерфейс задовольняє всім сучасним вимогам до розробки користувацького інтерфейсу.

# ВИСНОВКИ

Проведено аналіз наукових робіт та сучасних алгоритів розпізнавання емоцій на обличчі людини. Для аналізу були відібрані найпопулярніші та найефективніші алгоритми що застосовуються для розпізнавання.

За результатами аналізу були отримані наступні результати: алгоритм adaboost – 94% правильних відповідей; метод на основі текстурного аналізу – 93% правильних відповідей; метод на основі прихованих Марківських мереж – 92% правильних відповідей, багатошаровий персептрон 94%. Таким чином, можемо зробити висновок що найкраще за результатами дослідження пройшли системи на основі багатошарового персептрону та алгоритму adaboost.

Запропоновано розширення алгоритму AdaBoost і порівняно з сучасним алгоритмом виявлення обличчя Віоли та Джонса. Отримані результати в порівнянні з оригінальним алгоритмом AdaBoost з точки зору виявлення обличчя та розпізнавання емоцій кращі приблизно на 2,6%. Класифікатор, навчений за новим методом, був приблизно на 4% швидше, до того ж він складається лише з чверті слабких класифікаторів, необхідних для класифікатора, навченого стандартним AdaBoost. Алгоритм може бути застосований з іншими слабкими класифікаторами, придатними для виявлення обличчя. Досліджено та розроблено методику розпізнавання емоцій на обличчі людини у відеопотоці. На основі проведених досліджень запропоновано модифікацію алгоритму розпізнавання емоцій AdaBoost, з використанням ознак Хаара у якості набору класифікаторів. Встановлено відповідність між конкретною емоцією та регіоном на обличчі людини який найбільше відповідає за її вираження. Використовуючи запропоновану методику розроблено систему розпізнавання емоцій. Тестування системи проводились на наборі шести базових емоцій (радість, гнів, сум, огида, здивування, страх). В результаті отримали покращення точності розпізнавання на 2,6 %.

Проведено аналіз результатів дослідження, протестувано розроблену модель системи яка базується на зміненому алгоритмі розпізнавання емоцій на обличчі людини AdaBoost та ознаків Хаара. Зрівняли результати роботи звичайного алгоритму та зміненого. Згідно отриманого результату, в середньому розроблений алгоритм

верифікації працює з точністю розпізнавання 92.6% що знаходиться в межах заданої гарантії на початку 90% < 92.6% < 94%.

Підводячи підсумки можна сказати що система у 98% випадків які проводились під час тестування справляється з завдання окреслення області обличчя що є перешим етапом до розпізнавання емоцій. Що стосується розпізнавання емоцій в цілому то ситуація гірше, близько 93% випадків правильно розпізнаних емоціій.

Найкраще системі вдається розпізнавати такі емоції як радість, майже у 100% випадків вказана емоція була розпізнана правильно. Що стосується таких емоцій як гнів, огида, сум, здивування то вони займають середнє місце по складності розпізнавання. За їх участі траплялися помилки при розпізнаванні. Найважчою для розпізнавання виявилася емоція страх, вона була правильно розпізнане лише в 78% випадків.

Також була протестована працездатність програми за різних поворотів голови актора. Система коректно розпізнає емоції при повороті голови в праву чи ліву сторону на кут до 20 градусів. Гірша ситуація при нахилі голови вгору та вниз на 25-30 градусів, збільшується можливість помилки при розпізнавані. Також в процесі тестування виявлено серйозний недостаток, при нахилі голови в праву або ліву сторону система перестає не тільки окреслювати область обличчя так і взагалі розпізнавати емоції.

За результатами дослідження підготовлена наукова стаття для публікації у фаховому виданні.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Экман П. Психология эмоций. Я знаю, что ты чувствуешь / П. Экман. — Питер, 2010. — 336 с.

2. Khandait S. P. Automatic Facial Feature Extraction and Expression Recognition based on Neural Network / S. P. Khandait, R. C Thool, P. D. Khandait // International Journal of Advanced Computer Science and Applications. — 2011. — Vol. 2, No. 1. — P. 113–118.

3. Khandait S. P. ANFIS and BPNN based Expression Recognition using HFGA for Feature Extraction / S. P. Khandait, R. C. Thool, P. D. Khandait // Bulletin of Electrical Engineering and Informatics. — 2013. — Vol. 2, No. 1. — P. 11–22.

4. Gomathi V. Human Facial Expression Recognition Using MANFIS Model / V. Gomathi, K. Ramar, A. S. Jeevakumar // Proceedings of World Academy of Science Engineering and Technology. — 2009. — 38. — с. 338–342.

5. I. Cohen. Emotion Recognition from Facial Expressions Using Multilevel HMM / I. Cohen, A. Garg, T. S. Huang // Neural Information Processing Systems. — 2000. — с. 312–315.

6. Єфімов Г. М. Технологія для моделювання та розпізнавання емоційної міміки на обличчі людини / Г. М. Єфімов // Системи обробки інформації. — 2012. — с. 36–39.

7. Jang J.-S. R. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence / J.-S. R. Jang, C.-T. Sun, E. Mizutani. — Prentice Hall, 1997. — p. 614 .

8. Хайкин С. Нейронные сети: полный курс : пер. с англ. / С. Хайкин. — [2-е изд.]. — М. : Издательский дом «Вильямс». — 2006. — 1104 с.

9. N. Tsapatsoulis. A Fuzzy System for Emotion Classification Based on the MPEG-4 Facial Definition Parameter Set / N. Tsapatsoulis, K. Karpouzis, G. Stamou, F. Piat, S. Kollias // EURASIP Journal on Applied Signal Processing. — 2002. — p. 1021–1038.

10. Noldus Information Technology. Face Reader homepage [Електронний ресурс]. — Режим доступу: http://www.noldus.com/facereader/facereader-online

11. Visual Recognition. eMotion homepage [Електронний ресурс]. — Режим доступу: http://www.visual-recognition.nl/eMotion.html

12. Face Analysis and Emotion Recognition [Електронний ресурс]. — Режим доступу: http://www.amiproject.org/showcase/still-and-moving-image-processing/emotion-recognition

13. Affdex homepage [Електронний ресурс]. — Режим доступу: http://www.affdex.com/

14. Ekman P. Manual of the Facial Action Coding System (FACS) / P. Ekman, W. V. Friesen. — Palo Alto : Consulting Psychologists Press, 1978.

15. Ekman P. Facial Action Coding System: Investigator's Guide / P. Ekman, W. V. Friesen. — Palo Alto : Consulting Psychologists Press, 1978.

16. Artificial Intelligence and Human-Robot Interaction [Електронний ресурс]. — Режим доступу: http://www.affdex.com/

17. AdaBoost [Електронний ресурс] / wikipedia – стаття. – Режим доступу:https://ru.wikipedia.org/wiki/AdaBoost (

18. Facial expression recognition software FaceReader [Електронний ресурс] – Режим доступу: http://www.noldus.com/human-behaviorresearch/products/facereader

19. Нейроботикс - EmoDetect [Електронний ресурс]. – Режим доступу: http://neurobotics.ru/robotics/robotic-software/emodetect

20. Products - Cognitec // The face recognition company - [Електронний ресурс]– Режим доступу: http://www.cognitec.com/products.html

21. Microsoft Cognitive Services - Emotion API [Електронний ресурс] – Режим доступу: https://www.projectoxford.ai/demo/Emotion

22. Sirovich L. Low dimensional procedure for characterization of human faces/ L. Sirovich, M. Kirby – Journal of the Optical Society of America A, – 1987.– Vol. 4, P. 519.

23. Turk M. Face recognition using eigenfaces / M. Turk, A. Pentland – Proc. IEEE Conference on Computer Vision and Pattern Recognition. – 1991 – P. 586–591.

24. Метод главных компонент [Електронний ресурс] / wikipedia – стаття. – Режим доступу: https://ru.wikipedia.org/wiki/Метод_главных_компонент.

25. About OpenCV [Електронний ресурс] – Режим доступу: http://opencv.org/about.html.

26. Система кодирования лицевых движений [Електронний ресурс] / wikipedia – стаття. – Режим доступу: https://ru.wikipedia.org/wiki/Кодирования

27. Дэвид Форсайт, Жан Понс. Компьютерное зрение. Современный подход. 2004, 928 с.

28. S. Strupp Visual-Based Emotion Detection for Natural Man-Machine Interaction /Strupp S – 2008. –Vol. 5243, P.356 -363.

29. P. Viola Robust Real-time Object Detection / Viola P. M., Jones M.– 2001.

30. P. Viola Rapid object detection using a boosted cascade of simple features /Viola P., Jones M 2001. – p.324.

31. Трофимов Б.Ф. Методичні вказівки до оформлення програмної частини курсових та дипломних робіт. – Одеса, ОНПУ, 2012. [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/onputrpo2016/

32. Samaria F. Face Recognition Using Hidden Markov Models, PhD Thesis, University of Cambridge/ F. Samaria – 1994. – p.341.

33. Федоров М. Метод забезпечення інваріантності зображення обличчя щодо афінних спотворень/ М. Федоров – 2013. – № 3. – с. 294-298.

34. Marqu´es. Face Recognition Algorithms. Proyecto Fin de Carrera / Marqu´es – 2010. – p. 66.

35. Abate F. 2D and 3D face recognition: A survey. Pattern Recognition Letters / F. Abate – Vol.28. – 2007. – P.232.

36. Verma M. Comparison of Different Algorithms of Face Recognition. VSDR-IJEECE / M. Verma– Vol2(5). – 2012. – p. 272-278.

# ДОДАТОК А

# КЛЮЧОВІ ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

**File adaboost.py**

```
from config import DEBUG_MODEL
from config import USING_CASCADE

from config import LABEL_POSITIVE
from config import LABEL_NEGATIVE

from config import EXPECTED_TPR
from config import EXPECTED_FPR

from config import ROC_FILE

from weakClassifier import WeakClassifier
from matplotlib     import pyplot
from haarFeature    import Feature

import numpy
import time
import pylab


def getCachedAdaBoost(mat = None, label = None, filename = "", limit = 0):
    """
        Construct a AdaBoost object with cached data
        from file @ADABOOST_FILE """

    fileObj = open(filename, "a+")

    print "Constructing AdaBoost from existed model data"

    tmp = fileObj.readlines()

    if len(tmp) == 0:
        raise ValueError("There is no cached AdaBoost model")

    weakerNum = len(tmp) / 4
    model     = AdaBoost(train = False, limit = weakerNum)

    if limit < weakerNum:
        model.weakerLimit = limit
    else:
        model.weakerLimit = weakerNum

    for i in xrange(0, len(tmp), 4):

        alpha, dimension, direction, threshold = None, None, None, None

        for j in xrange(i, i + 4):
            if   (j % 4) == 0:
                alpha     = float(tmp[j])
            elif (j % 4) == 1:
                dimension = int(tmp[j])
            elif (j % 4) == 2:
                direction = float(tmp[j])
            elif (j % 4) == 3:
                threshold = float(tmp[j])

        classifier = model.Weaker(train = False)
  classifier.constructor(dimension, direction, threshold)
  classifier._mat = mat
  classifier._label = label
```

```
            if mat is not None:
                classifier.sampleNum = mat.shape[1]

            model.G[i/4]     = classifier
            model.alpha[i/4] = alpha
            model.N         += 1

        model._mat = mat
        model._label = label
        if model.N > limit:
            model.N    = limit

        if label is not None:
            model.samplesNum = len(label)

        print "Construction finished"
        fileObj.close()

        return model


class AdaBoost:
    """
        Parameter:
        @Mat    :   A matrix(or two dimension array) which's size is
                    (row    = number of features,
                    column  = number of total sample)
        @Tag    :   A vector(or one dimension array) which's size is the
                    same as the number of total sample
        @classifier: Object. A instance of weaker classifier.

        @train  :   A bool value. If it's False, it means that user want to
                    get a instance of this class object from cached data
        @limit  :   A integer. The limitation of training times."""


    def __init__(self, Mat = None, Tag = None, classifier = WeakClassifier, train = True, limit = 4):
        if train == True:
            self._mat   = Mat
            self._label = Tag

            self.samplesDim, self.samplesNum = self._mat.shape

            # Make sure that the inputted data's dimension is right.
            assert self.samplesNum == self._label.size

            self.posNum = numpy.count_nonzero(self._label == LABEL_POSITIVE)
            self.negNum = numpy.count_nonzero(self._label == LABEL_NEGATIVE)

            # Initialization of weight
            pos_W = [1.0/(2 * self.posNum) for i in range(self.posNum)]

            neg_W = [1.0/(2 * self.negNum) for i in range(self.negNum)]
            self.W = numpy.array(pos_W + neg_W)

            self.accuracy = []

        self.Weaker = classifier

    self.weakerLimit = limit

    self.G      = [None for _ in xrange(limit)]
    self.alpha  = [  0  for _ in xrange(limit)]
    self.N      = 0
    self.detectionRate = 0.

    # true positive rate
```

```
    self.tpr = 0.
    # false positive rate
    self.fpr = 0.

    self.th  = 0.


def is_good_enough(self):

    output = self.prediction(self._mat, self.th)

    correct = numpy.count_nonzero(output == self._label)/(self.samplesNum*1.)
    self.accuracy.append( correct)

    self.detectionRate = numpy.count_nonzero(output[0:self.posNum] == LABEL_POSITIVE) * 1./ self.posNum

    Num_tp = 0 # Number of true positive
    Num_fn = 0 # Number of false negative
    Num_tn = 0 # Number of true negative
    Num_fp = 0 # Number of false positive
    for i in xrange(self.samplesNum):
        if self._label[i] == LABEL_POSITIVE:
            if output[i] == LABEL_POSITIVE:
                Num_tp += 1
            else:
                Num_fn += 1
        else:
            if output[i] == LABEL_POSITIVE:
                Num_fp += 1
            else:
                Num_tn += 1

    self.tpr = Num_tp * 1./(Num_tp + Num_fn)
    self.fpr = Num_fp * 1./(Num_tn + Num_fp)

    if self.tpr > EXPECTED_TPR and self.fpr < EXPECTED_FPR:
        return True

def train(self):
    """
    function @train() is the main process which run
    AdaBoost algorithm."""

    adaboost_start_time = time.time()

    for m in xrange(self.weakerLimit):
        self.N += 1

        if DEBUG_MODEL == True:
            weaker_start_time = time.time()

        self.G[m] = self.Weaker(self._mat, self._label, self.W)

        errorRate = self.G[m].train()

        if DEBUG_MODEL == True:
            print "Time for training WeakClassifier:", \
                  time.time() - weaker_start_time

        if errorRate < 0.0001:
            errorRate = 0.0001

        beta = errorRate / (1 - errorRate)
        self.alpha[m] = numpy.log(1/beta)

        output = self.G[m].prediction(self._mat)
```

```
        for i in xrange(self.samplesNum):
            #self.W[i] *= numpy.exp(-self.alpha[m] * self._label[i] * output[i])
            if self._label[i] == output[i]:
                self.W[i] *=  beta

        self.W /= sum(self.W)

        if USING_CASCADE is True:
            self.th, self.detectionRate = self.findThreshold(EXPECTED_TPR)

        if self.is_good_enough():
            print (self.N) ," weak classifier is enough to ",
            print "meet the request which given by user."
            print "Training Done :)"
            break

        if DEBUG_MODEL is True:
            print "weakClassifier:", self.N
            print "errorRate     :", errorRate
            print "accuracy      :", self.accuracy[-1]
            print "detectionRate :", self.detectionRate
            print "AdaBoost's Th :", self.th
            print "alpha         :", self.alpha[m]

    #self.showErrRates()
    #self.showROC()

    print "The time cost of training this AdaBoost model:",\
            time.time() - adaboost_start_time

    output = self.prediction(self._mat, self.th)
    return output, self.fpr


def grade(self, Mat):

    #Mat = numpy.array(Mat)

    sampleNum = Mat.shape[1]

    output = numpy.zeros(sampleNum, dtype = numpy.float16)

    for i in xrange(self.N):
        output += self.G[i].prediction(Mat) * self.alpha[i]

    return output


def prediction(self, Mat, th = None):

    #Mat = numpy.array(Mat)

    output = self.grade(Mat)

    if th == None:
        th = self.th

    """
    # Don't do this! Bug!! the first statement will rewrite the output
    output[output > th]  = LABEL_POSITIVE
    output[output <= th] = LABEL_NEGATIVE
    """

    for i in range(len(output)):
        if output[i] > th:
            output[i] = LABEL_POSITIVE
        else:
```

```
            output[i] = LABEL_NEGATIVE

        return output


    def findThreshold(self, expected_tpr):
        detectionRate = 0.
        best_th       = None

        low_bound = -sum(self.alpha)
        up__bound = +sum(self.alpha)
        step      = -0.1
        threshold = numpy.arange(up__bound - step, low_bound + step, step)

        for t in xrange(threshold.size):

            output = self.prediction(self._mat, threshold[t])

            Num_tp = 0 # Number of true positive
            Num_fn = 0 # Number of false negative
            Num_tn = 0 # Number of true negative
            Num_fp = 0 # Number of false positive
            for i in range(self.samplesNum):
                if self._label[i] == LABEL_POSITIVE:
                    if output[i] == LABEL_POSITIVE:
                        Num_tp += 1
                    else:
                        Num_fn += 1
                else:
                    if output[i] == LABEL_POSITIVE:
                        Num_fp += 1
                    else:
                        Num_tn += 1

            tpr = Num_tp * 1./(Num_tp + Num_fn)
            fpr = Num_fp * 1./(Num_tn + Num_fp)

            if tpr >= expected_tpr:

                detectionRate = numpy.count_nonzero(output[0:self.posNum] == LABEL_POSITIVE) * 1./ self.posNum

                best_th = threshold[t]
                break

        return best_th, detectionRate

    def showErrRates(self):

        pyplot.title("The changes of accuracy (Figure by Jason Leaster)")
        pyplot.xlabel("Iteration times")
        pyplot.ylabel("Accuracy of Prediction")
        pyplot.plot([i for i in xrange(self.N)],
                    self.accuracy, '-.',
                    label = "Accuracy * 100%")
        pyplot.axis([0., self.N, 0, 1.])

        if DEBUG_MODEL == True:
            pyplot.show()
        else:
            pyplot.savefig("accuracyflow.jpg")

    def showROC(self):
        best_tpr = 0.
        best_fpr = 1.
        best_th  = None

        low_bound = -sum(self.alpha) * 0.5
```

```
    up__bound = +sum(self.alpha) * 0.5
    step      = 0.1
    threshold = numpy.arange(low_bound, up__bound, step)

    tprs      = numpy.zeros(threshold.size, dtype = numpy.float16)
    fprs      = numpy.zeros(threshold.size, dtype = numpy.float16)

    for t in xrange(threshold.size):

        output = self.prediction(self._mat, threshold[t])

        Num_tp = 0 # Number of true positive
        Num_fn = 0 # Number of false negative
        Num_tn = 0 # Number of true negative
        Num_fp = 0 # Number of false positive
        for i in range(self.samplesNum):
            if self._label[i] == LABEL_POSITIVE:
                if output[i] == LABEL_POSITIVE:
                    Num_tp += 1
                else:
                    Num_fn += 1
            else:
                if output[i] == LABEL_POSITIVE:
                    Num_fp += 1
                else:
                    Num_tn += 1

        tpr = Num_tp * 1./(Num_tp + Num_fn)
        fpr = Num_fp * 1./(Num_tn + Num_fp)

        # if tpr >= best_tpr and fpr <= best_fpr:
        #     best_tpr = tpr
        #     best_fpr = fpr
        #     best_th  = threshold[t]

        tprs[t] = tpr
        fprs[t] = fpr

    fileObj = open(ROC_FILE, "a+")
    for t, f, th in zip(tprs, fprs, threshold):
        fileObj.write(str(t) + "\t" + str(f) + "\t" + str(th) + "\n")

    fileObj.flush()
    fileObj.close()

    pyplot.title("The ROC curve")
    pyplot.plot(fprs, tprs, "-r", linewidth = 1)
    pyplot.xlabel("fpr")
    pyplot.ylabel("tpr")
    pyplot.axis([-0.02, 1.1, 0, 1.1])
    if DEBUG_MODEL == True:
        pyplot.show()
    else:
        pyplot.savefig("roc.jpg")

def saveModel(self, filename):
    """
        function @saveModel save the key data member of AdaBoost
    into a template file @ADABOOST_FILE
    """
    fileObj = open(filename, "a+")

    for m in xrange(self.N):
        fileObj.write(str(self.alpha[m]) + "\n")
        fileObj.write(str(self.G[m].opt_dimension) + "\n")
        fileObj.write(str(self.G[m].opt_direction) + "\n")
        fileObj.write(str(self.G[m].opt_threshold) + "\n")
```

```
    fileObj.flush()
    fileObj.close()

def makeClassifierPic(self):
    from config import TRAINING_IMG_HEIGHT
    from config import TRAINING_IMG_WIDTH
    from config import WHITE
    from config import BLACK
    from config import FIGURES

    from config import HAAR_FEATURE_TYPE_I
    from config import HAAR_FEATURE_TYPE_II
    from config import HAAR_FEATURE_TYPE_III
    from config import HAAR_FEATURE_TYPE_IV
    from config import HAAR_FEATURE_TYPE_V

    IMG_WIDTH  = TRAINING_IMG_WIDTH
    IMG_HEIGHT = TRAINING_IMG_HEIGHT

    haar = Feature(IMG_WIDTH, IMG_HEIGHT)

    featuresAll = haar.features
    selFeatures = [] # selected features

    for n in xrange(self.N):
        selFeatures.append(featuresAll[self.G[n].opt_dimension])

    classifierPic = numpy.zeros((IMG_HEIGHT, IMG_WIDTH))

    for n in xrange(self.N):
        feature   = selFeatures[n]
        alpha     = self.alpha[n]
        direction = self.G[n].opt_direction

        (types, x, y, width, height) = feature

        image = numpy.array([[155 for i in xrange(IMG_WIDTH)] for j in xrange(IMG_HEIGHT)])

        assert x >= 0 and x < IMG_WIDTH
        assert y >= 0 and y < IMG_HEIGHT
        assert width > 0 and height > 0

        if direction == +1:
            black = BLACK
            white = WHITE
        else:
            black = WHITE
            white = BLACK

        if types == HAAR_FEATURE_TYPE_I:
            for i in xrange(y, y + height * 2):
                for j in xrange(x, x + width):
                    if i < y + height:
                        image[i][j] = black
                    else:
                        image[i][j] = white

        elif types == HAAR_FEATURE_TYPE_II:
            for i in xrange(y, y + height):
                for j in xrange(x, x + width * 2):
                    if j < x + width:
                        image[i][j] = white
                    else:
                        image[i][j] = black

        elif types == HAAR_FEATURE_TYPE_III:
```

```
            for i in xrange(y, y + height):
                for j in xrange(x, x + width * 3):
                    if j >= (x + width) and j < (x + width * 2):
                        image[i][j] = black
                    else:
                        image[i][j] = white

        elif types == HAAR_FEATURE_TYPE_IV:
            for i in xrange(y, y + height*3):
                for j in xrange(x, x + width):
                    if i >= (y + height) and i < (y + height * 2):
                        image[i][j] = black
                    else:
                        image[i][j] = white

        elif types == HAAR_FEATURE_TYPE_V:
            for i in xrange(y, y + height * 2):
                for j in xrange(x, x + width * 2):
                    if (j < x + width and i < y + height) or\
                       (j >= x + width and i >= y + height):
                        image[i][j] = white
                    else:
                        image[i][j] = black
        else:
            raise Exception("Unkown type feature")

        #classifierPic += image * alpha * direction
        classifierPic += image


        pyplot.matshow(image, cmap = "gray")
        if DEBUG_MODEL == True:
            pylab.show()
        else:
            pyplot.savefig(FIGURES + "feature_" + str(n) + ".jpg")

    from image import Image
    classifierPic = Image._normalization(classifierPic)
    pylab.matshow(classifierPic, cmap = "gray")
    if DEBUG_MODEL == True:
        pylab.show()
    else:
        pyplot.savefig(FIGURES + "boosted_features.jpg")
```

**file Cascade.py**

```
from config   import POSITIVE_SAMPLE
from config   import NEGATIVE_SAMPLE
from config   import TRAINING_IMG_HEIGHT
from config   import TRAINING_IMG_WIDTH
from config   import FEATURE_FILE_TRAINING
from config   import FEATURE_NUM
from config   import ADABOOST_LIMIT
from config   import ADABOOST_CACHE_FILE
from config   import DEBUG_MODEL

from haarFeature import Feature
from image       import ImageSet
from adaboost    import AdaBoost
from adaboost    import getCachedAdaBoost

import os
import numpy


class Cascade:
```

```
def __init__(self, face_dir = "", nonface_dir = "", train = True, limit = 30):
    #tot_samples = 0

    self.Face    = ImageSet(face_dir,    sampleNum = POSITIVE_SAMPLE)
    self.nonFace = ImageSet(nonface_dir, sampleNum = NEGATIVE_SAMPLE)

    tot_samples = self.Face.sampleNum + self.nonFace.sampleNum

    self.classifier = AdaBoost

    self.haar   = Feature(TRAINING_IMG_WIDTH, TRAINING_IMG_HEIGHT)

    if os.path.isfile(FEATURE_FILE_TRAINING + ".npy"):

        self._mat = numpy.load(FEATURE_FILE_TRAINING + ".npy")

    else:
        if DEBUG_MODEL is True:
            self._mat = numpy.zeros((self.haar.featuresNum, tot_samples))

            for i in xrange(self.Face.sampleNum):
                featureVec = self.haar.calFeatureForImg(self.Face.images[i])
                for j in xrange(self.haar.featuresNum):
                    self._mat[j][i                    ]  = featureVec[j]

            for i in xrange(self.nonFace.sampleNum):
                featureVec = self.haar.calFeatureForImg(self.nonFace.images[i])
                for j in xrange(self.haar.featuresNum):
                    self._mat[j][i + self.Face.sampleNum] = featureVec[j]

            numpy.save(FEATURE_FILE_TRAINING, self._mat)
        else:
            from mapReduce import map
            from mapReduce import reduce

            map(self.Face, self.nonFace)
            self._mat = reduce()

    featureNum, sampleNum = self._mat.shape

    assert sampleNum  == (POSITIVE_SAMPLE + NEGATIVE_SAMPLE)
    assert featureNum == FEATURE_NUM

    Label_Face    = [+1 for i in xrange(POSITIVE_SAMPLE)]
    Label_NonFace = [-1 for i in xrange(NEGATIVE_SAMPLE)]

    self._label = numpy.array(Label_Face + Label_NonFace)
    self.limit  = limit
    self.classifierNum     = 0
    self.strong_classifier = [None for i in xrange(limit)]


def train(self):

    raise ("Unfinished")

    detection_rate = 0
    from config import EXPECTED_FPR_PRE_LAYYER
    from config import EXPECTED_FPR
    from config import LABEL_NEGATIVE

    cur_fpr = 1.0
    mat   = self._mat
    label = self._label

    for i in xrange(self.limit):
```

```
        if cur_fpr < EXPECTED_FPR:
            break
        else:
            cache_filename = ADABOOST_CACHE_FILE + str(i)

            if os.path.isfile(cache_filename):
                self.strong_classifier[i] = getCachedAdaBoost(mat     = self._mat,
                                                              label   = self._label,
                                                              filename= cache_filename,
                                                              limit   = ADABOOST_LIMIT)
            else:
                self.strong_classifier[i] = AdaBoost(mat, label, limit = ADABOOST_LIMIT)
                output, fpr = self.strong_classifier[i].train()

                cur_fpr *= fpr

                fp_num = fpr * numpy.count_nonzero(label == LABEL_NEGATIVE)

                self.strong_classifier[i].saveModel(cache_filename)
                mat, label = self.updateTrainingDate(mat, output, fp_num)

            self.classifierNum += 1


    def updateTrainingDate(self, mat, output, fp_num):

        fp_num = int(fp_num)

        assert len(output) == self._label.size

        _mat = numpy.zeros((FEATURE_NUM, POSITIVE_SAMPLE + fp_num), dtype=numpy.float16)

        _mat[:, :POSITIVE_SAMPLE] = mat[:, :POSITIVE_SAMPLE]
        """
        for i in xrange(POSITIVE_SAMPLE):
            for j in xrange(FEATURE_NUM):
                mat[j][i] = self._mat[j][i]
        """

        counter = 0
        # only reserve negative samples which are classified wrong
        for i in xrange(POSITIVE_SAMPLE, self._label.size):
            if output[i] != self._label[i]:
                for j in xrange(FEATURE_NUM):
                    _mat[j][POSITIVE_SAMPLE + counter] = mat[j][i]
                counter += 1

        assert counter == fp_num

        Label_Face    = [+1 for i in xrange(POSITIVE_SAMPLE)]
        Label_NonFace = [-1 for i in xrange(fp_num)]

        _label = numpy.array(Label_Face + Label_NonFace)

        return _mat, _label


    def predict(self):

        output = numpy.zeros(POSITIVE_SAMPLE + NEGATIVE_SAMPLE, dtype= numpy.float16)
        for i in xrange(self.classifierNum):

            self.strong_classifier[i].prediction(mat, th = 0)

            """unfinished"""

    def save(self):
```

```
        pass

    def is_goodenough(self):
        pass

    /**
     * The method scroll down the page in order to trigger all the js units
     * @param driver
     */
    public void scrollPageDown(WebDriver driver) {
        try{
            ((JavascriptExecutor) driver).executeScript("window.scrollTo(0, document.body.scrollHeight–5)");
            Thread.sleep(2000);
            ((JavascriptExecutor) driver).executeScript("window.scrollTo(0, –document.body.scrollHeight–5)");
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

**File HaarFeature.py**

```
"""
    For each feature pattern, the start point(x, y) is at
    the most left-up pixel in that window. The size of that
    window is @width * @height
"""
import numpy

from config import HAAR_FEATURE_TYPE_I
from config import HAAR_FEATURE_TYPE_II
from config import HAAR_FEATURE_TYPE_III
from config import HAAR_FEATURE_TYPE_IV
from config import HAAR_FEATURE_TYPE_V

from image import Image


class Feature:
    def __init__(self, img_Width, img_Height):

        self.featureName = "Haar Feature"

        self.img_Width  = img_Width
        self.img_Height = img_Height

        self.tot_pixels = img_Width * img_Height

        self.featureTypes = (HAAR_FEATURE_TYPE_I,
                             HAAR_FEATURE_TYPE_II,
                             HAAR_FEATURE_TYPE_III,
                             HAAR_FEATURE_TYPE_IV,
                             HAAR_FEATURE_TYPE_V)

        self.features    = self._evalFeatures_total()

        self.featuresNum = len(self.features)

        #self.featureMat  =     numpy.zeros((self.tot_pixels, self.featuresNum),
        #                        dtype=numpy.float16)

        # just for running faster and save RAM. allocate once and use many times.
        self.vector          = numpy.zeros(self.featuresNum, dtype=numpy.float32)

        self.idxVector_tmp_0 = numpy.zeros(self.tot_pixels, dtype = numpy.int8)
        self.idxVector_tmp_1 = numpy.zeros(self.tot_pixels, dtype = numpy.int8)
        self.idxVector_tmp_2 = numpy.zeros(self.tot_pixels, dtype = numpy.int8)
```

```
    self.idxVector_tmp_3 = numpy.zeros(self.tot_pixels, dtype = numpy.int8)


def vecRectSum(self, idxVector, x, y, width, height):
    idxVector *= 0 # reset this vector
    if x == 0 and y == 0:
        idxVector[width * height + 2] = +1

    elif x == 0:
        idx1 = self.img_Height * (    width - 1) + height + y - 1
        idx2 = self.img_Height * (    width - 1) +          y - 1
        idxVector[idx1] = +1
        idxVector[idx2] = -1

    elif y == 0:
        idx1 = self.img_Height * (x + width - 1) + height - 1
        idx2 = self.img_Height * (x         - 1) + height - 1
        idxVector[idx1] = +1
        idxVector[idx2] = -1
    else:
        idx1 = self.img_Height * (x + width - 1) + height + y - 1
        idx2 = self.img_Height * (x + width - 1) +          y - 1
        idx3 = self.img_Height * (x         - 1) + height + y - 1
        idx4 = self.img_Height * (x         - 1) +          y - 1

        assert idx1 < self.tot_pixels and idx2 < self.tot_pixels
        assert idx3 < self.tot_pixels and idx4 < self.tot_pixels

        idxVector[idx1] = + 1
        idxVector[idx2] = - 1
        idxVector[idx3] = - 1
        idxVector[idx4] = + 1

    return idxVector


def VecFeatureTypeI(self, vecImg, x, y, width, height):
    vec1 = self.vecRectSum(self.idxVector_tmp_0, x, y         , width, height)
    vec2 = self.vecRectSum(self.idxVector_tmp_1, x, y + height, width, height)

    featureSize = width * height * 2

    return (vec1.dot(vecImg) - vec2.dot(vecImg))/featureSize


def VecFeatureTypeII(self, vecImg, x, y, width, height):
    vec1 = self.vecRectSum(self.idxVector_tmp_0, x + width, y, width, height)
    vec2 = self.vecRectSum(self.idxVector_tmp_1, x        , y, width, height)

    featureSize = width * height * 2

    return (vec1.dot(vecImg) - vec2.dot(vecImg))/featureSize


def VecFeatureTypeIII(self,vecImg, x, y, width, height):
    vec1 = self.vecRectSum(self.idxVector_tmp_0, x +   width, y, width, height)
    vec2 = self.vecRectSum(self.idxVector_tmp_1, x          , y, width, height)
    vec3 = self.vecRectSum(self.idxVector_tmp_2, x + 2*width, y, width, height)

    featureSize = width * height * 3

    return (vec1.dot(vecImg) - vec2.dot(vecImg)
          - vec3.dot(vecImg))/featureSize


def VecFeatureTypeIV(self,vecImg, x, y, width, height):
    vec1 = self.vecRectSum(self.idxVector_tmp_0, x, y +   height, width, height)
```

```
    vec2 = self.vecRectSum(self.idxVector_tmp_1, x, y          , width, height)
    vec3 = self.vecRectSum(self.idxVector_tmp_2, x, y + 2*height, width, height)

    featureSize = width * height * 3

    return (vec1.dot(vecImg) - vec2.dot(vecImg)
          - vec3.dot(vecImg))/featureSize


def VecFeatureTypeV(self, vecImg, x, y, width, height):
    vec1 = self.vecRectSum(self.idxVector_tmp_0, x + width,         y, width, height)
    vec2 = self.vecRectSum(self.idxVector_tmp_1, x       ,         y, width, height)
    vec3 = self.vecRectSum(self.idxVector_tmp_2, x       , y + height, width, height)
    vec4 = self.vecRectSum(self.idxVector_tmp_3, x + width, y + height, width, height)

    featureSize = width * height * 4

    return (vec1.dot(vecImg) - vec2.dot(vecImg) +
          vec3.dot(vecImg) - vec4.dot(vecImg))/featureSize


def _evalFeatures_total(self):
    win_Height = self.img_Height
    win_Width  = self.img_Width

    height_Limit = {HAAR_FEATURE_TYPE_I    : win_Height/2 - 1,
                    HAAR_FEATURE_TYPE_II   : win_Height   - 1,
                    HAAR_FEATURE_TYPE_III  : win_Height   - 1,
                    HAAR_FEATURE_TYPE_IV   : win_Height/3 - 1,
                    HAAR_FEATURE_TYPE_V    : win_Height/2 - 1}

    width_Limit  = {HAAR_FEATURE_TYPE_I   : win_Width   - 1,
                    HAAR_FEATURE_TYPE_II  : win_Width/2 - 1,
                    HAAR_FEATURE_TYPE_III : win_Width/3 - 1,
                    HAAR_FEATURE_TYPE_IV  : win_Width   - 1,
                    HAAR_FEATURE_TYPE_V   : win_Width/2 - 1}

    features = []
    for types in self.featureTypes:
        for w in xrange(1, width_Limit[types]):
            for h in xrange(1, height_Limit[types]):

                if w == 1 and h == 1:
                    continue

                if types == HAAR_FEATURE_TYPE_I:

                    x_limit = win_Width  - w
                    y_limit = win_Height - 2*h
                    for x in xrange(1, x_limit):
                        for y in xrange(1, y_limit):
                            features.append( (types, x, y, w, h))

                elif types == HAAR_FEATURE_TYPE_II:
                    x_limit = win_Width  - 2*w
                    y_limit = win_Height - h
                    for x in xrange(1, x_limit):
                        for y in xrange(1, y_limit):
                            features.append( (types, x, y, w, h))

                elif types == HAAR_FEATURE_TYPE_III:
                    x_limit = win_Width  - 3*w
                    y_limit = win_Height - h
                    for x in xrange(1, x_limit):
                        for y in xrange(1, y_limit):
                            features.append( (types, x, y, w, h))
```

```
            elif types == HAAR_FEATURE_TYPE_IV:
                x_limit = win_Width  - w
                y_limit = win_Height - 3*h
                for x in xrange(1, x_limit):
                    for y in xrange(1, y_limit):
                        features.append( (types, x, y, w, h))

            elif types == HAAR_FEATURE_TYPE_V:
                x_limit = win_Width  - 2*w
                y_limit = win_Height - 2*h
                for x in xrange(1, x_limit):
                    for y in xrange(1, y_limit):
                        features.append( (types, x, y, w, h))

    return features


def calFeatureForImg(self, img):

    assert isinstance(img, Image)
    assert img.img.shape[0] == self.img_Height
    assert img.img.shape[1] == self.img_Width

    for i in xrange(self.featuresNum):
        type, x, y, w, h = self.features[i]

        if   type == HAAR_FEATURE_TYPE_I:
            self.vector[i] = self.VecFeatureTypeI(img.vecImg, x, y, w, h)
        elif type == HAAR_FEATURE_TYPE_II:
            self.vector[i] = self.VecFeatureTypeII(img.vecImg, x, y, w, h)
        elif type == HAAR_FEATURE_TYPE_III:
            self.vector[i] = self.VecFeatureTypeIII(img.vecImg, x, y, w, h)
        elif type == HAAR_FEATURE_TYPE_IV:
            self.vector[i] = self.VecFeatureTypeIV(img.vecImg, x, y, w, h)
        elif type == HAAR_FEATURE_TYPE_V:
            self.vector[i] = self.VecFeatureTypeV(img.vecImg, x, y, w, h)
        else:
            raise Exception("unknown feature type")

    return self.vector


def makeFeaturePic(self, feature):

    from matplotlib import pyplot
    from config     import BLACK
    from config     import WHITE
    import pylab

    (types, x, y, width, height) = feature

    assert x >= 0 and x < self.img_Width
    assert y >= 0 and y < self.img_Height
    assert width > 0 and height > 0

    image = numpy.array([[125. for i in xrange(self.img_Width)]
                         for j in xrange(self.img_Height)])

    if types == HAAR_FEATURE_TYPE_I:
        for i in xrange(y, y + height * 2):
            for j in xrange(x, x + width):
                if i < y + height:
                    image[i][j] = BLACK
                else:
                    image[i][j] = WHITE
```

```
    elif types == HAAR_FEATURE_TYPE_II:
        for i in xrange(y, y + height):
            for j in xrange(x, x + width * 2):
                if j < x + width:
                    image[i][j] = WHITE
                else:
                    image[i][j] = BLACK

    elif types == HAAR_FEATURE_TYPE_III:
        for i in xrange(y, y + height):
            for j in xrange(x, x + width * 3):
                if j >= (x + width) and j < (x + width * 2):
                    image[i][j] = BLACK
                else:
                    image[i][j] = WHITE

    elif types == HAAR_FEATURE_TYPE_IV:
        for i in xrange(y, y + height*3):
            for j in xrange(x, x + width):
                if i >= (y + height) and i < (y + height * 2):
                    image[i][j] = BLACK
                else:
                    image[i][j] = WHITE

    elif types == HAAR_FEATURE_TYPE_V:
        for i in xrange(y, y + height * 2):
            for j in xrange(x, x + width * 2):
                if (j < x + width and i < y + height) or\
                   (j >= x + width and i >= y + height):
                    image[i][j] = BLACK
                else:
                    image[i][j] = WHITE


    pyplot.matshow(image, cmap = "gray")
    pylab.show()


"""
old version. Don't use this.
def _evalFeatures(self):
    win_Height = self.img_Height
    win_Width  = self.img_Width

    height_Limit = {HAAR_FEATURE_TYPE_I   : win_Height/2 - 1,
                    HAAR_FEATURE_TYPE_II  : win_Height   - 1,
                    HAAR_FEATURE_TYPE_III : win_Height   - 1,
                    HAAR_FEATURE_TYPE_IV  : win_Height/2 - 1}

    width_Limit  = {HAAR_FEATURE_TYPE_I   : win_Width   - 1,
                    HAAR_FEATURE_TYPE_II  : win_Width/2 - 1,
                    HAAR_FEATURE_TYPE_III : win_Width/3 - 1,
                    HAAR_FEATURE_TYPE_IV  : win_Width/2 - 1}

    features = []
    for types in self.featureTypes:
        for w in xrange(1, width_Limit[types]):
            for h in xrange(1, height_Limit[types]):

                y_start = None

                if types == HAAR_FEATURE_TYPE_I:
                    x_limit = win_Width  - w
                    y_limit = win_Height - 2*h
                    for x in xrange(1, x_limit):
                        for y in xrange(1, y_limit, 2):
                            features.append( (types, x, y, w, h))
```

```
elif types == HAAR_FEATURE_TYPE_II:
    x_limit = win_Width  - 2*w
    y_limit = win_Height - h
    for x in xrange(1, x_limit):
        if h % 2 == 1:
            if x % 2 == 1:
                y_start = 1
            else:
                y_start = 2
        else:
            y_start = 1

        for y in xrange(y_start, y_limit, 2):
            features.append( (types, x, y, w, h))

elif types == HAAR_FEATURE_TYPE_III:
    x_limit = win_Width  - 3*w
    y_limit = win_Height - h
    for x in xrange(1, x_limit):
        if w == 1:
            if h % 2 == 1:
                if (h + 1)/2 % 2 == 1:
                    if x % 2 == 1:
                        y_start = 1
                    else:
                        y_start = 2
                else:
                    if x % 2 == 1:
                        y_start = 2
                    else:
                        y_start = 1
            else:
                if (h/2) % 2 == 1:
                    y_start = 2
                else:
                    y_start = 1
        elif w == 2:
            if h % 2 == 1:
                if x % 2 == 1:
                    y_start = 2
                else:
                    y_start = 1
            else:
                y_start = 2

        elif w == 3:
            if h % 2 == 1:
                if (h+1)/2 % 2 == 1:
                    if x % 2 == 1:
                        y_start = 2
                    else:
                        y_start = 1
                else:
                    if x % 2 == 1:
                        y_start = 1
                    else:
                        y_start = 2
            else:
                if (h/2) % 2 == 1:
                    y_start = 1
                else:
                    y_start = 2
        #elif w == 4:
        else:
            if h % 2 == 1:
                if x % 2 == 1:
```

```
                        y_start = 1
                    else:
                        y_start = 2
                else:
                    y_start = 1

                for y in xrange(y_start, y_limit, 2):
                    features.append( (types, x, y, w, h))

            elif types == HAAR_FEATURE_TYPE_IV:
                x_limit = win_Width  - 2*w
                y_limit = win_Height - 2*h
                for x in xrange(1, x_limit):
                    for y in xrange(1, y_limit, 2):
                        features.append( (types, x, y, w, h))
    return features
"""
```

**file weekClasifier.py**

```
from matplotlib import pyplot
import numpy

from config import LABEL_POSITIVE
from config import LABEL_NEGATIVE


class WeakClassifier:

    def __init__(self, Mat = None, Tag = None, W = None, train = True):
        """
        Parameter:
        @Mat    :   A matrix(or two dimension array) which's size is
                    (row    = number of features,
                    column  = number of total sample)
        @Tag    :   A vector(or one dimension array) which's size is the
                    same as the number of total sample

        @W      :   Weight of each sample in the training set.
                    A vector or a list, which's size is the same as the
                    number of total sample.

        @train  :   A bool value. If it's False, it means that user want to
                    get a instance of this class object from cached data"""

        if train == True:
            """
            It's necessary to do this check.
            The implementation depend on numpy.ndarray heavily
            """
            assert Mat.__class__ == numpy.ndarray
            assert Tag.__class__ == numpy.ndarray
            assert   W.__class__ == numpy.ndarray

            """
            It will cost a lot of memory, if I use @Mat to initialize
            the @self._mat like this:
                self._mat = numpy.array(Mat)

            constructor @numpy.array will return a new object which's
            message is the same as @Mat

            To save memory, I just set the data member @self._mat
            the same as the parameter passed into this constructor,
            which means that they point to the same address.

            Make sure this weak classifier will not modify the inputed mat.
```

```
        """
        self._mat = Mat
        self._label = Tag

        # sampleDim == the number of features
        self.sampleDim, self.sampleNum = self._mat.shape

        if W == None:
            self.numPos = numpy.count_nonzero(self._label == LABEL_POSITIVE)
            self.numNeg = numpy.count_nonzero(self._label == LABEL_NEGATIVE)
            pos_W = [1.0/(2 * self.numPos) for i in xrange(self.numPos)]

            neg_W = [1.0/(2 * self.numNeg) for i in xrange(self.numNeg)]
            self.weight = numpy.array(pos_W + neg_W)

        else:
            self.weight = W

        self.output = numpy.zeros(self.sampleNum, dtype = numpy.int)

        self.opt_errorRate = 1.
        self.opt_dimension = 0
        self.opt_threshold = None
        self.opt_direction = 0


def optimal(self, d):

    # for positive sample
    idx = (self._label + LABEL_POSITIVE) / (LABEL_POSITIVE * 2)
    weight = self.weight  * idx
    vector = self._mat[d] * idx
    sumPos = weight.dot(vector)
    sumPosW= weight.sum()

    # for negative sample
    idx = (self._label + LABEL_NEGATIVE) / (LABEL_NEGATIVE * 2)
    weight = self.weight  * idx
    vector = self._mat[d] * idx
    sumNeg = weight.dot(vector)
    sumNegW= weight.sum()

    """
    Code beyong there is just optimal version of this one.
    ======================================================
    sumPos = 0.
    sumNeg = 0.

    sumPosW = 0.
    sumNegW = 0.

    for i in xrange(self.sampleNum):
        if self._label[i] == LABEL_POSITIVE:
            sumPos  += self.weight[i] * self._mat[d][i]
            sumPosW += self.weight[i]
        else:
            sumNeg  += self.weight[i] * self._mat[d][i]
            sumNegW += self.weight[i]
    """

    miuPos = sumPos / sumPosW
    miuNeg = sumNeg / sumNegW

    threshold = (miuPos + miuNeg)/2

    minErrRate    = numpy.inf
    bestDirection = None
```

```
    for direction in [-1, 1]:
        errorRate = 0.

        self.output[self._mat[d] * direction < threshold * direction]\
                = LABEL_POSITIVE

        self.output[self._mat[d] * direction >= threshold * direction]\
                = LABEL_NEGATIVE

        errorRate = self.weight[ self.output != self._label].sum()

        self.output *= 0 # reset the output
        start = time.time()
        for i in xrange(self.sampleNum):
            if self._mat[d][i] *direction < threshold * direction:
                self.output[i] = LABEL_POSITIVE
            else:
                self.output[i] = LABEL_NEGATIVE

            if self.output[i] != self._label[i]:
                errorRate += self.weight[i]
        """

        self.output *= 0 # reset the output
        if errorRate < minErrRate:
            minErrRate    = errorRate
            bestDirection = direction

    return minErrRate, threshold, bestDirection

def train(self):

    for dim in xrange(self.sampleDim):
        err, threshold, direction = self.optimal(dim)
        if err < self.opt_errorRate:
            self.opt_errorRate = err
            self.opt_dimension = dim
            self.opt_threshold = threshold
            self.opt_direction = direction

    assert self.opt_errorRate < 0.5

    return self.opt_errorRate

def prediction(self, Mat):
    sampleNum = Mat.shape[1]

    dim       = self.opt_dimension
    threshold = self.opt_threshold
    direction = self.opt_direction

    output = numpy.zeros(sampleNum, dtype = numpy.int)

    output[Mat[dim] * direction <  direction * threshold] = LABEL_POSITIVE
    output[Mat[dim] * direction >= direction * threshold] = LABEL_NEGATIVE
    """
    for i in xrange(sampleNum):
        if direction * Mat[dim][i] < direction * threshold:
            output[i] = LABEL_POSITIVE
        else:
            output[i] = LABEL_NEGATIVE
    """

    return output

def show(self, dim = None):
```

```
if dim == None:
   dim = self.opt_dimension

N = 10 # the number of center
MaxVal = numpy.max(self._mat[dim])
MinVal = numpy.min(self._mat[dim])

scope = (MaxVal - MinVal) / N

centers = [ (MinVal - scope/2)+ scope*i for i in xrange(N)]
counter = [ [0, 0] for i in xrange(N)]

for j in xrange(N):
   for i in xrange(self.sampleNum):
      if abs(self._mat[dim][i] - centers[j]) < scope/2:
         if self._label[i] == LABEL_POSITIVE:
            counter[j][1] += 1
         else:
            counter[j][0] += 1

posVal, negVal = [], []

for i in xrange(N):
   posVal.append(counter[i][1])
   negVal.append(counter[i][0])

sumPosVal = sum(posVal)
sumNegVal = sum(negVal)

for i in xrange(len(posVal)): posVal[i] /= (1. * sumPosVal)
for i in xrange(len(negVal)): negVal[i] /= (1. * sumNegVal)

pyplot.title("A simple weak classifier")
pyplot.plot(centers, posVal, "r-o", label = "Face class")
pyplot.plot(centers, negVal, "b-o", label = "Non-Face class")
pyplot.xlabel("feature response")
pyplot.ylabel("frequency")

# plot threshold line
sumPosW = 0.
sumNegW = 0.
sumPos = 0.
sumNeg = 0.
for i in xrange(self.sampleNum):
   if self._label[i] == LABEL_POSITIVE:
      sumPos  += self.weight[i] * self._mat[dim][i]
      sumPosW += self.weight[i]
   else:
      sumNeg  += self.weight[i] * self._mat[dim][i]
      sumNegW += self.weight[i]
```

# ДОДАТОК Б

УДК 004.051

## ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ ЕМОЦІЙ ЛЮДИНИ ЗА ЗОБРАЖЕННЯМ ОБЛИЧЧЯ

**Стеценко І.В., Яковлев Д.П.**

## RESEARCH OF HUMAN EMOTIONS RECOGNITION METHODS BEHIND THE FACE IMAGE

**Stetsenko I., Yakovlev D.**

*Дослідження присвячене методам розпізнавання емоцій людини за зображенням обличчя. Досліджено сучасні алгоритми, що використовуються для розпізнавання базових емоцій людини за фронтальним статичним зображенням її обличчя. Обґрунтовано вибір методу с найбільш високою точністю. Визначено вектори ознак для обраного класифікатора. Проведено тестування системи на тестових наборах зображень обличь людей і реальних випадкових даних для основних емоцій (гнів, відраза, страх, радість, сум, здивування). Запропонована система визначає емоцію людини з середньою точністю до 95,4%.*

***Ключові слова:*** *розпізнавання емоцій, нейронні мережі, метод головних компонент, еластичні порівняння на графах.*

**Вступ.** В даний час бурхливо розвивається галузь робототехніки що займається створенням мобільних роботів. Дедалі популярнішими стають сервісні мобільні роботи. До даного типу можна віднести роботів-екскурсоводів, роботів-пилососів, рухливі інформаційні термінали, безпілотні автомобілі, тощо. Для виконання своїх функцій ці роботи повинні вміти розпізнавати та класифікувати об’єкти навколишнього середовища в автоматичному режимі.

У наш час інформаційні технології досягли значного розвитку й продовжують розвиватися. Разом із тим, підвищується актуальність задач розпізнавання та класифікації образів.

Автоматичне розпізнавання та аналіз виразів обличчя є активною темою в наукових дослідженнях вже більше двох десятиліть [1]. Нещодавні психологічні дослідження показали, що міміка є найбільш точним способом виявлення емоцій у людей. Словесна частина повідомлення дорівнює лише 7% ефекту повідомлення в цілому, вокальна частина - 38%, а вираз обличчя - 55% від ефекту повідомлення спікера [2].

Тому автоматичне розпізнавання обличчя та його емоціонального виразу в режимі реального часу може бути корисним для багатьох галузей, наприклад віртуальної реальності, відеоконференцій, дослідження задоволеності клієнтів тощо. Інформаційна система, що сприймає людські емоції, забезпечує ліпший інтерфейс взаємодії робота з людиною, адаптується під її настрій і стан. Емоції можуть стати додатковим невербальним фільтром під час пошуку інформації, бути причиною зміни ігрового, або тренувального процесу, чи братись до уваги під час оцінки істинності вербальної інформації з боку роботизованої системи.

Саме розпізнавання людських емоцій може стати основною частиною проекту, де на додаток до біометричних параметрів оточення робота, додається інформація про міміку, що грає важливу роль при вирішенні такого завдання, як розпізнати настрій людини. Хоча для людини досить просто виявляти та розуміти емоційний стан інших людей, проте точність розпізнавання емоцій машиною залишається проблемою.

На сьогоднішній день розроблено низку комерційних програмних систем, які розпізнають емоції за зображенням обличчя. Проте, вони мають певні обмеження, такі як незадовільну похибку розпізнавання, або ж дуже високу вартість, тому актуальність розпізнавання емоцій людини на обличчі людини залишається на досить високому рівні.

**Мета дослідження.** Знайти, обґрунтувати, та вибрати алгоритм розпізнавання базових емоцій людини за фронтальним статичним зображенням її обличчя, який надає високу точність з малими витратами обчислювальних ресурсів з можливостю використання їх у автономних мобільних системах.

Завданнями дослідження є:

- аналітичний огляд сучасного стану вирішення проблеми розпізнавання емоцій людини;
- розробка алгоритму визначення емоції людини;
- розробка моделі системи для визначення емоції людини;
- аналіз результатів моделювання;.

**Аналіз сучасного стану вирішення проблеми визначення емоцій людини.** На сьогоднішній день існує велика кількість методів виявлення обличь. Виділяють чотири категорії методів виявлення обличь (рис.1).

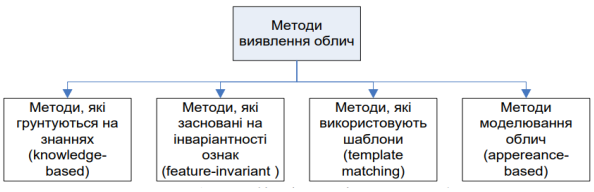


Рис. 1. Класифікація методів виявлення обличь

Перші три категорії методів роблять спробу визначити і використати принципи, якими керується мозок при вирішенні задачі виявлення обличчя. Методи перших трьох категорій, як правило, використовуються для локалізації обличчя на зображеннях високої якості. Ці методи стійкі до різних умов освітлення, але мають високу обчислювальну вартість [3]. На відміну від перших трьох категорій методи, що ґрунтуються на моделюванні обличчя підходять до проблеми виявлення обличь з іншого боку. Ці методи не намагаються у явному вигляді формалізувати процеси, що відбуваються в людському мозку, а намагаються виявити закономірності і методи виявлення обличь.

Методи, які засновані на інваріантності ознак (feature-invariant) властивості зображення обличчя неявно, застосовуючи методи математичної статистики і машинного навчання. Методи цієї категорії спираються на інструментарій розпізнавання образів, розглядаючи задачу виявлення обличчя, як окремий випадок задачі розпізнавання. Вони беруть за основу статистичні методи навчання, щоб побудувати класифікатор «обличчя»/«не обличчя» з навчальних прикладів. Ці методи, як правило, використовуються для виявлення обличь на зображеннях не дуже високої роздільної здатності [4].

Зі швидким збільшенням обчислювальних ресурсів та пам’яті комп’ютерів методи моделювання обличчя стали домінувати при створенні детекторів обличь. Загальна практика полягає у зборі великого набору прикладів «обличь» і «не обличь», та застосування певних алгоритмів машинного навчання, щоб навчити модель обличчя здійснювати бінарну класифікацію [5].

Існує велика кількість методів фільтрації зображень у складних умовах освітлення для систем відеоспостереження. Усі ці підходи для покращення зображень поділяється на дві великі категорії: методи оброблення у просторовій області і методи оброблення у частотній області. Категорія, методів оброблення у просторовій області, поєднує підходи, засновані на прямому маніпулюванні пікселями зображення.

Методи оброблення в частотній області ґрунтуються на модифікації сигналу, сформованого шляхом застосування до зображення перетворення Фур'є. Також не є другорядними і технології, що базуються на різних комбінаціях методів з цих двох категорій [6]. При роботі в темний час доби ефективність системи відеоспостереження знижується за рахунок того, що різко зростає шум сигналу, який веде до того, що на зображенні з'являється зернистість. А це призводить до того, що відбувається збільшення бітрейту з причини поганого освітлення, і, отже, виходить погане ущільнення відзнятого матеріалу [7]. Нейронні мережі можуть бути з успіхом застосовані для виявлення обличь. При цьому якість розпізнавання знижується при збільшенні кількості класів, які необхідно передбачити. Перевагою використання штучних нейронних мереж для виявлення обличь є можливість навчання системи для виділення ключових характеристик обличь на навчальних вибірках.

Нейронні мережі забезпечують можливість одержання класифікатора добре моделюючого складну функцію розподілу зображень обличь. У завданнях класифікації при цьому відбувається неявне виділення ключових ознак усередині мережі, визначення значимості ознак і системи взаємних залежностей між ними. Серед нейронних мереж для розв'язку завдань розпізнавання образів найчастіше застосовуються багатошарові персептрони зі зворотним поширенням помилки, мережі з радіально-базисною функцією й згорточні нейронні мережі (Convolutional Neural Networks) [8].

**Дослідження методів розпізнавання емоцій людини.** Дослідження проводилося серед найпопулярніших та найефективніших алгоритмів, що застосовуються для розпізнавання:

– метод на основі текстурного аналізу;
– багатошаровий персептрон;
– алгоритм AdaBoost;
– приховані Марківські моделі.

Гістограма спрямованих градієнтів (англ. Histogram of Oriented Gradients, HOG) - дескриптори особливих точок, які використовуються в комп'ютерному зорі і обробці зображень з метою розпізнавання об'єктів. Дана техніка заснована на підрахунку кількості напрямків градієнта в локальних областях зображення. Цей метод схожий на гістограми напрямки краю, дескриптори SIFT і контексти форми, але відрізняється тим, що обчислюється на щільної сітці рівномірно розподілених осередків і

використовує нормалізацію перекривати локального контрасту для збільшення точності.

Основною ідеєю алгоритму є припущення, що зовнішній вигляд і форма об'єкта на ділянці зображення можуть бути описані розподілом градієнтів інтенсивності або напрямком країв. Реалізація цих дескрипторів може бути проведена шляхом поділу зображення на маленькі пов'язані області, іменовані осередками, і розрахунком для кожного осередку гістограми напрямків градієнтів або напрямків країв для пікселів, що знаходяться всередині осередку. Комбінація цих гістограм і є дескриптором.

Для збільшення точності локальні гістограми піддаються нормалізації по контрасту. З цією метою обчислюється міра інтенсивності на більшому фрагменті зображення, який називається блоком, і отримане значення використовується для нормалізації. Нормалізовані дескриптори мають кращу інваріантність по відношенню до висвітлення [9].

Дескриптор HOG має кілька переваг над іншими дескрипторами. Оскільки HOG працює локально, метод підтримує інваріантність геометричних і фотометричних перетворень, за винятком орієнтації об'єкта. Подібні зміни з'являться тільки в великих фрагментах зображення. Більш того, грубе розбиття простору, точне обчислення напрямків і сильна локальна фотометрична нормалізація дозволяють ігнорувати рухи пішоходів, якщо вони підтримують вертикальне положення тіла. Дескриптор HOG, таким чином, є хорошим засобом знаходження обличь на зображеннях.

Першим кроком обчислень у багатьох детекторах особливих точок є нормалізація кольору і гамма-корекція. Дослідження показали, що для дескриптора HOG цей крок можна опустити, оскільки подальша нормалізація дасть той же результат. Тому на першому етапі розраховуються значення градієнтів. Найпоширенішим методом є застосування одновимірної диференційної маски в горизонтальному або вертикальному напрямку. Цей метод вимагає фільтрації колірної або складової яскравості за допомогою наступних фільтруючих ядер:

$$[-1,0,1] \text{ і } [-1,0,1]^T \qquad (1)$$

Використання більш складних масок, такі як Собел 3x3 (Оператор Собеля) або діагональні маски показали нижчу продуктивність для даного завдання [10]. Використання розмивання по Гаусу перед застосуванням диференціальної маски, зменшує швидкодію без помітного покращення якості.

На наступному кроці обчислюються гістограми осередків. Кожен піксель в осередку бере участь в підвішеному голосуванні для каналів гістограми напрямків, заснованому на значенні градієнтів. Осередки можуть бути прямокутної або круглої форми, канали гістограми рівномірно розподіляються від 0 до 180 або ж від 0 до 360 градусів, в залежності від того, обчислюється «знаковий» або «беззнаковий градієнт».

Для прийняття до уваги яскравості і контрастності градієнти локально нормалізують, для чого осередки згруповують в більш великі зв'язуючі блоки. Дескриптор HOG, таким чином, є вектором компонент нормованих гістограм осередків з усіх областей блоку. Як правило, блоки перекриваються, тобто кожен осередок входить більш ніж в один кінцевий дескриптор. Використовуються дві основні геометрії блоку: прямокутні R-HOG і круглі C-HOG. Блоки R-HOG зазвичай є квадратними сітками, що характеризуються трьома параметрами: кількістю осередків на блок, кількістю пікселів на осередок і кількістю каналів на гістограму осередка.

Наявна кількість різноманітних підходів класифікації образів вимагає окреслення переваг використання саме багатошарового персептрону (рис.2).

Як нейронна мережа персептрон характеризується такими властивостями, як "Відображення вхідної інформації у вихідну", "Адаптивність", "Очевидність відповіді", "Відмовостійкість", "Ефективність реалізації на НВІС", "Аналогія з нейробіологією" та інші. Для навчання багатошарового персептрону використовується алгоритм зворотного поширення помилки.

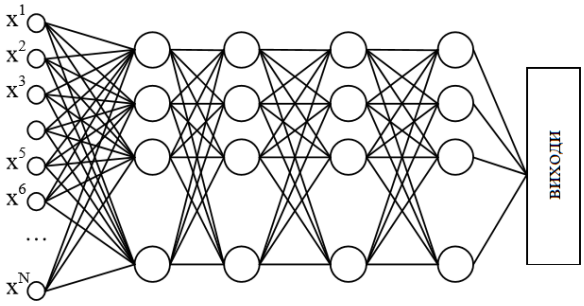


Рис. 2. Архітектура багатошарового персептрону

Архітектура системи розпізнавання складається із шести підсистем, в яких відбуваються певні етапи обробки вхідного зображення (рис. 3).

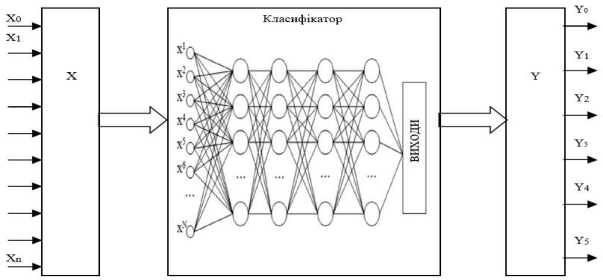


Рис. 3. Архітектура системи розпізнавання

Підсистема розпізнавання емоцій складається із наступних частин (рис. 4):


Рис. 4. Підсистема розпізнавання емоцій

Вхідний образ $X$ являє собою відносні координати $(x, y)$ елементів обличчя із початком відліку в центрі обличчя, а також множину сигналів із пороговими значеннями 0 або 1, що відповідають відсутності чи присутності зморшок на чолі, біля очей, на носі, на підборідді. Так корисна інформація потрапляє в нейронну мережу, яка далі класифікує розпізнану емоцію;

Класифікатор на основі багатошарового персептрону має від 1 до 3 шарів. Вихідний шар містить 6 нейронів, які відповідають за кожну базову емоцію. Значення вихідних сигналів у межах [0, 1], де 0 означає, що ознак базової емоції не знайдено, а 1 – що базова емоція присутня на зображенні у найбільш явному вигляді (чітко виражена);

Вихідний сигнал – це значення вихідного вектора персептрону, що відповідають за кожну базову емоцію: гнів, відраза, страх, радість, сум, здивування.

Передові дослідження в галузі психології довели, що емоції всіх людей мають спільні зовнішні прояви у міміці людей. Це дає підстави розглядати можливість створення універсального класифікатора емоцій. У роботі запропоновано підхід до розпізнавання мімічних проявів емоцій людини із використанням багатошарового персептрону, який має такі переваги:

– висока точність розпізнавання;
– висока швидкодія;
– адаптація до змін вхідних образів;
– низький рівень споживання ресурсів.

Однак лише за мімічними проявами неможливо із 100% ймовірністю виявити присутність тієї чи іншої емоції. Для вирішення цієї проблеми пропонується інтегрувати цей класифікатор у систему поряд із класифікаторами, які аналізують голос, пантоміміку, зміни фізіологічних процесів в організмі тощо.

Одним з статистичних методів розпізнавання осіб є приховані моделі Маркова (ПММ) з дискретним часом. ПММ використовують статистичні властивості сигналів і враховують безпосередньо їх просторові характеристики. Елементами моделі є: множина прихованих станів, множина спостережуваних станів, матриця перехідних ймовірностей, початкова ймовірність станів. Кожному відповідає своя модель Маркова. При розпізнаванні об'єкта перевіряються згенеровані для заданої бази об'єктів моделей Маркова і шукається максимальна із спостережуваних ймовірність того, що послідовність спостережень для даного об'єкта згенерована відповідною моделлю.

Для вирішення цього завдання необхідно виконати два етапи: виявити обличчя на фотографії і розпізнати його. Елементами ПММ є множина спостережуваних символів (спостереження в даному випадку - це коефіцієнти дискретного косинусного перетворення фотографії обличчя людини скануючим вікном певного розміру), множина різних станів, набір спостережуваних символів, вектор початкових станів, матриця перехідних ймовірностей, матриця ймовірностей спостережуваних символів.

Можливість застосування ПММ до задачі розпізнавання осіб пов'язано з тим, що:

– в наявної послідовності псевдовипадкових змінних (фрагментів обличчя) ми можемо вважати, що кожне спостереження є незалежним від попередніх;

– кожна випадкова змінна дає вимір, розподіл ймовірностей яких залежить від стану.

Завдання розпізнавання ділиться на дві підзадачі: на етапі навчання ПММ треба побудувати модель по набору різних зображень особи конкретної людини, що зберігаються в базі даних; на етапі розпізнавання деяке запропоноване зображення слід віднести до однієї з моделі з будь-якої ймовірністю (рис. 5).

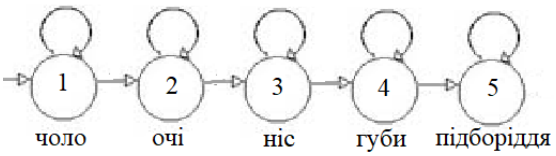


Рис. 5. Приклад марківського ланцюга одновимірної ПММ.

На базі даних Olivetti Research Ltd був досягнутий середній відсоток розпізнавання близько 90%, максимальний відсоток розпізнавання склав 95%. ORL - база даних, що складається з 400 напівтонових зображень 40 людей - співробітників Olivetti Research Ltd і студентів Кембриджського університету.

Деякі співвідношення параметрів моделі призводять до збільшення відсотка розпізнавання (наприклад, при 85% перетину параметрів скануючого вікна). Ускладнення моделі певним компромісним варіантом між одновимірною ПММ і умовно двовимірною ПММ реалізується зменшенням скануючого вікна по ширині і просуванням його по змієподібній траєкторії: зліва на право і зверху вниз (рис. 6), [11].

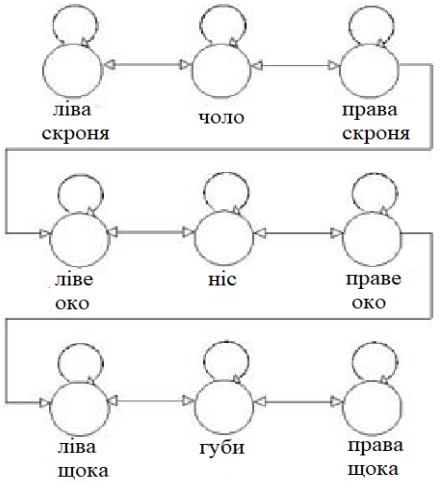


Рис. 6. Приклад марківського ланцюга ускладненої одномірної ПММ

На тій же базі даних було досягнуто середній відсоток розпізнавання в 85% і максимальний в 96.5%. Падіння числа вірно розпізнаних зображень (в середньому) пов'язано, по всій ймовірності з тим, що в результаті роботи алгоритму визначення оптимальної послідовності станів відбувалося тільки в горизонтальному напрямку, а у вертикальному напрямку послідовності станів ініціювалася тільки на початковому етапі.

Для адекватної оцінки впливу інтенсивності пікселів в області обличчя на точність розпізнавання емоцій необхідно в першу чергу визначити базовий алгоритм, з яким буде проводитися порівняння. У даній роботі в якості базового використовується один з варіантів класичного алгоритму на основі положення ключових точок. По суті, цей алгоритм розділений на два етапи: отримання координат ключових точок і класифікація емоцій на їх основі.

Класифікація об'єктів по вектору ознак (в даному випадку координат точок) є стандартною задачею навчання з учителем і не представляє інтересу. Слід лише зазначити, що використовувався один з найбільш популярних класифікаторів, а саме метод опорних векторів (англ. Support vector machines, SVM).

Найбільш популярний метод використовується у так званих активні моделі зовнішнього вигляду (англ. Active appearance models, AAM) [12]. Вхідними даними для методу є набір зображень обличь, в якому кожному зображенню відповідає файл розмітки, що містить координати ключових точок, обраних людиною. За цими даними AAM будує дві статистичні моделі: модель форми - параметричну лінійну модель, що описує можливі варіації положення ключових точок. Формою при цьому називається вектор координат ключових точок - подібну модель, але описує вже можливі варіації інтенсивності пікселів. Відповідно текстурою називається вектор всіх пікселів всередині зовнішнього контуру форми [13].

В цьому підході інформація про текстуру використовується для отримання координат ключових точок, але на етап класифікації вона не передається.

Можливість визначення емоцій за матеріальним становищем і формі ключових елементів обличчя добре вивчена як психологами, так і фахівцями в галузі інформатики. У той же час аналогічні судження щодо текстури практично не робилися. Існує можливість визначення (вручну або комп'ютерними засобами) виразу обличчя людини навіть за відсутності інформації про становище і формі його ключових елементів. Іншими словами, передбачається, що емоції на обличчі кодуються не тільки положенням ключових точок, а й іншими ознаками.

Як і в методі на основі положення ключових точок, перетворення до усередненої формі дозволяє сформувати для кожного зображення відповідний вектор текстури.

Цей вектор можна безпосередньо використовувати в якості вектора ознак при класифікації. В якості класифікатора можна застосовувати SVM. В результаті отримуємо наступний метод розпізнавання емоцій на основі інформації про текстуру. На етапі навчання:

– зображення піддаються попередній обробці, і т. д. За допомогою активних моделей зовнішнього вигляду на них визначаються положення ключових точок, а потім кусочно-афінним перетворенням особи на зображеннях приводяться до усередненої формі;

– пікселі отриманих зображень разом з відповідними позначками емоцій використовуються для навчання методом SVM;

На етапі застосування:

– за аналогією зі стадією навчання нові зображення приводяться до середньої формі;

– перетворені зображення класифікуються за допомогою навченої раніше моделі. Забігаючи наперед, скажемо, що точність розпізнавання методом на основі текстури лише трохи поступається точності класичного підходу.

Підводячи проміжний підсумок, підкреслимо ще раз спільні та відмінні риси двох методів. Обидва включають етапи навчання (створення моделі) і застосування. Обидва вимагають попередньої обробки зображень для отримання ознак. Перший метод використовує в якості ознак координати ключових точок (форму), отриманих за допомогою моделей активного способу, другий - текстуру зображення всередині зовнішнього контуру обличчя. В якості кінцевого класифікатора в обох описаних методах використовується SVM. Для приведення обличь до єдиної формі застосовується активна модель зовнішнього вигляду, проте це не є обов'язковою вимогою. Для аналогічних цілей може бути використаний пошук кількох ключових точок і подальше 3D-моделювання [14].

Хоча описані вище методи дають досить високі результати самі по собі, можливо об'єднати

їх, поєднавши обидві набори ознак: координати ключових точок і інтенсивності пікселів в області обличчя. Оскільки теоретичне обґрунтування для всіх частин алгоритму вже було дано в попередніх розділах, тут просто опишемо послідовність кроків комбінованого методу розпізнавання емоцій. На етапі навчання:

– до вхідних зображень застосовуються активні моделі образу для отримання координат ключових точок. Вектори ключових точок для кожного зображення організовуються в матрицю S;

– до зображень застосовується кусочно-афінне перетворення для приведення їх до середньої форми: для отриманих на попередньому кроці точок будується тріангуляція Делоне, а потім пікселі всередині кожного окремого трикутника відображаються на відповідні пікселі всередині середньої форми.

– пікселі кожного перетвореного зображення організовуються в єдиний вектор ознак, а вектори для всіх зображень організовуються в матрицю T;

– матриці S і T об'єднуються так, що кожен вектор s з матриці S розширюється відповідним вектором із матриці T. Іншими словами, ознаки з двох наборів об'єднуються в єдиний вектор. Результуючу матрицю назвемо X;

– мітки емоцій об'єднуються в єдиний вектор y;

– матриця даних X і вектор міток y використовуються для навчання моделі SVM.

Як і передбачалося, такий комбінований підхід дав точність, що перевершує результати обох описаних раніше методів.

**Результати дослідження.** Порівнюючи показники AAM та SVM, можна сказати, що на стадії застосування AAM визначає положення ключових точок в середньому за 180-190 мс, а SVM проводить класифікацію приблизно за 25 мс, що робить можливим використання розроблених алгоритмів в системах реального часу. Точність розпізнавання визначалася стандартним методом перехресної перевірки. При цьому для навчання класифікатора завжди використовувалося останнє зображення (особа, максимально сильно виражає емоцію) в кожній з послідовностей кадрів (всього 327 зображень).

За результатами аналізу були отримані наступні результати: алгоритм adaboost – 94% правильних відповідей; метод на основі текстурного аналізу – 93% правильних відповідей; метод на основі прихованих Марківських мереж – 92% правильних відповідей, багатошаровий персептрон 94% (рис. 7).

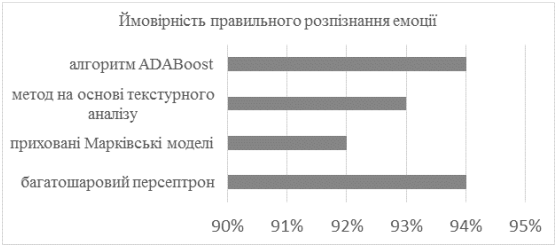


Рис. 7. Результати порівняльного аналізу алгоритмів розпізнавання

Таким чином, можемо зробити висновок що найкраще за результатами дослідження пройшли системи на основі багатошарового персептрону та алгоритму adaboost.

**Література**

1. Экман П. Психология эмоций. Я знаю, что ты чувствуешь / П. Экман. — Питер, 2010. — 336 с.
2. Khandait S. P. Automatic Facial Feature Extraction and Expression Recognition based on Neural Network / S. P. Khandait, R. C Thool, P. D. Khandait // International Journal of Advanced Computer Science and Applications. — 2011. — Vol. 2, No. 1. — P. 113–118.
3. Face Analysis and Emotion Recognition [Електронний ресурс]. — Режим доступу: http://www.amiproject.org/showcase/still-and-moving-image-processing/emotion-recognition.
4. Affdex homepage [Електронний ресурс]. — Режим доступу: http://www.affdex.com/
5. Ekman P. Manual of the Facial Action Coding System (FACS) / P. Ekman, W. V. Friesen. — Palo Alto : Consulting Psychologists Press, 1978.
6. Ekman P. Facial Action Coding System: Investigator's Guide / P. Ekman, W. V. Friesen. — Palo Alto : Consulting Psychologists Press, 1978.
7. Artificial Intelligence and Human-Robot Interaction [Електронний ресурс]. — Режим доступу: http://www.affdex.com/.
8. AdaBoost [Електронний ресурс] / wikipedia – стаття. – Режим доступу: https://ru.wikipedia.org/wiki/AdaBoost
9. Facial expression recognition software FaceReader [Електронний ресурс] – Режим доступу: http://www.noldus.com/human-behaviorresearch/products/facereader
10. Нейроботикс - EmoDetect [Електронний ресурс]. – Режим доступу: http://neurobotics.ru/robotics/robotic-software/emodetect
11. Microsoft Cognitive Services - Emotion API [Електронний ресурс] – Режим доступу: https://www.projectoxford.ai/demo/Emotion.
12. Sirovich L. Low dimensional procedure for characterization of human faces/ L. Sirovich, M. Kirby – Journal of the Optical Society of America A, – 1987.– Vol. 4, P. 519.
13. Turk M. Face recognition using eigenfaces / M. Turk, A. Pentland – Proc. IEEE Conference on Computer Vision and Pattern Recognition. – 1991 – P. 586–591.
14. Метод главных компонент [Електронний ресурс] / wikipedia – стаття. – Режим доступу: https://ru.wikipedia.org/wiki/Метод_главных_компонент.

**References**

1. Jekman P. Psihologija jemocij. Ja znaju, chto ty chuvstvuesh' / P. Jekman. — Piter, 2010. — 336 s.

2. Khandait S. P. Automatic Facial Feature Extraction and Expression Recognition based on Neural Network / S. P. Khandait, R. C Thool, P. D. Khandait // International Journal of Advanced Computer Science and Applications. — 2011. — Vol. 2, No. 1. — P. 113–118.
3. Face Analysis and Emotion Recognition [Elektronnij resurs]. — Rezhim dostupu http://www.amiproject.org/showcase/still-and-moving-image-processing/emotion-recognition.
4. Affdex homepage [Elektronnij resurs]. — Rezhim dostupu: http://www.affdex.com/
5. Ekman P. Manual of the Facial Action Coding System (FACS) / P. Ekman, W. V. Friesen. — Palo Alto : Consulting Psychologists Press, 1978.
6. Ekman P. Facial Action Coding System: Investigator's Guide / P. Ekman, W. V. Friesen. — Palo Alto : Consulting Psychologists Press, 1978.
7. Artificial Intelligence and Human-Robot Interaction [Elektronnij resurs]. — Rezhim dostupu: http://www.affdex.com/.
8. AdaBoost [Elektronnij resurs]. — Rezhim dostupu: https://ru.wikipedia.org/wiki/AdaBoost
9. Facial expression recognition software FaceReader [Elektronnij resurs]. — Rezhim dostupu: http://www.noldus.com/human-behaviorresearch/products/facereader
10. Neurobiotics - EmoDetect [Elektronnij resurs]. — Rezhim dostupu: http://neurobotics.ru/robotics/robotic-software/emodetect
11. Microsoft Cognitive Services - Emotion API [Elektronnij resurs]. — Rezhim dostupu https://www.projectoxford.ai/demo/Emotion.
12. Sirovich L. Low dimensional procedure for characterization of human faces/ L. Sirovich, M. Kirby – Journal of the Optical Society of America A, – 1987.– Vol. 4, P. 519.
13. Turk M. Face recognition using eigenfaces / M. Turk, A. Pentland – Proc. IEEE Conference on Computer Vision and Pattern Recognition. – 1991 – P. 586–591.
14. Metod glavnyh komponent [Elektronnij resurs]. — Rezhim dostupu: https://ru.wikipedia.org/wiki/Метод_главных_компонент.

**Стеценко И.В., Яковлев Д.П. Исследование методов распознования эмоций человека по изображению лица.**

*Исследование посвящено методам распознавания эмоций человека по изображению лица. Исследованы современные алгоритмы используемые для распознавания базовых эмоций человека по фронтальным статическим изображениям его лица. Обоснован выбор метода с наиболее высокой точностью. Определены векторы признаков для выбранного классификатора. Проведено тестирование системы на тестовых наборах изображений лиц людей и реальных случайных данных для основных эмоций (гнев, отвращение, страх, радость, грусть, удивление). Предложенная система определяет эмоцию человека со средней точностью 95,4 %.*

***Ключевые слова:*** *распознавание эмоций, нейронные сети, метод главных компонент, эластичные сравнения на графах.*

**Stetsenko I.V., Yakovlev D.P. Research of human emotions recognition methods behind the face image.**

*The study is devoted to methods of recognizing human emotions by facial images. Modern algorithms used to recognize the basic emotions of a person by the frontal static image of his face are studied. The choice of the method with the highest accuracy is substantiated. The feature vectors for the selected classifier are defined. The system was tested on test sets of images of people's faces and real random data for basic emotions (anger, disgust, fear, joy, sadness, surprise). The proposed system determines human emotion with an average accuracy of 95.4%.*

***Keywords:*** *emotion recognition, neural networks, principal components method, elastic comparisons on graphs.*

**Стеценко І.В.** – бакалавр, студент кафедри комп'ютерних систем Одеського національного політехнічного університету, e-mail: stetsenko.i.ks@gmail.com.

**Яковлев Д.П.** – к.т.н., проф. кафедри комп'ютерних систем Одеського національного політехнічного університету, e-mail: dpyakovlev39@gmail.com.

*Рецензент:* д.т.н., проф. **Ситніков В.С.**