

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ЗАКЛАД
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

Цінь Лу

ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ ГРАФІЧНИХ
ОБ'ЄКТІВ

Магістерська робота
за спеціальністю 122 "Комп'ютерні науки"

Особистий підпис – _____

Науковий керівник – _____ доцент, к.т.н. Галина КОЗУБ

В.о.зав. кафедри – _____ професор, д.т.н. Юрій КОЗУБ

Полтава – 2025

АНОТАЦІЯ

Цінь Лу

Тема: Дослідження методів розпізнавання графічних об'єктів.

Спеціальність: 122 „Комп'ютерні науки”.

Установа: ДЗ ЛНУ імені Тараса Шевченка, 2021р.

Кваліфікаційна робота містить: 71 стор., 29 рис., 39 джерел, 2 додатки.

Об'єкт дослідження – методи розпізнавання графічних образів.

Предмет дослідження – технології створення додатку для розпізнавання графічних об'єктів.

Мета роботи – програмна реалізація детектора маски для обличчя засобами Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask.

Результати роботи. Досліджено методи розпізнавання графічних образів та проблеми класифікації графічних образів. Розглянуто рецепторну структуру сприйняття інформації та алгоритми побудови моделі розпізнавання образів.

Розроблено вебдодаток відеоспостереження за визначенням маски для обличчя, з використанням сучасних засобів розробки Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask.

Ключові слова: Python, Tensorflow Keras, API, Numpy, OpenCV, Flask, вейвлет.

ABSTRACT

Qin Lu

Theme: Study of graphic object recognition methods.

Specialty: 122 "Computer Science".

Institution: Taras Shevchenko National University of Luhansk, 2020.

Qualification work contains: 71 pages, 29 figures, 39 sources, 2 appendices.

Object of research methods of graphic image recognition.

Subject of research - technologies for creating an application for recognizing graphic objects

Purpose of the study software implementation of a face mask detector using Python, OpenCV, Keras / TensorFlow, Deep Learning and Flask.

Results of research. Methods of graphic image recognition and problems of graphic image classification are investigated. The receptor structure of information perception and algorithms for building a pattern recognition model are considered. A web application for video surveillance for the definition of a face mask has been developed, using modern development tools Python, OpenCV, Keras / TensorFlow, Deep Learning and Flask.

Keywords: Python, Tensorflow Keras, API, Numpy, OpenCV, Flask, wavelet.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ	5
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	8
1.1. Методи розпізнавання графічних образів	10
1.2. Штучні нейронні мережі	18
1.2.1. Навчання нейронної мережі	20
1.3. Висновки до розділу 1	23
РОЗДІЛ 2. ПРОБЛЕМИ КЛАСИФІКАЦІЇ ГРАФІЧНИХ ОБРАЗІВ.....	24
2.1. Рецепторна структура сприйняття інформації.....	24
2.2. Алгоритмічні побудови	35
2.3. Модель розпізнавання образів.....	44
РОЗДІЛ 3. РОЗРОБКА ВЕБДОДАТКА ДЕТЕКТОРА МАСОК.....	50
3.1. Підготовка набору даних та навчання ML моделі.....	52
3.2. Розгортання застосунку та приклади роботи.....	57
3.3. Висновки до розділу 3	59
ВИСНОВКИ.....	60
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61
ДОДАТКИ.....	65
Додаток А. Код app.py	65
Додаток Б. Код mask.py	68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ

AI	-	artificial intelligence;
BFS	-	Breadth First Search;
BSP	-	Binary Space Partitioning;
GUI	-	graphical user interface;
HDFS	-	Heuristic Depth First Search;
LTV	-	lifetime value;
ML	-	machine learning;
OBT	-	Open beta testing;
SURF	-	Speeded-Up Robust Features;
SIFT	-	Scale-invariant Feature Transform;
HOG	-	Histogram of Oriented Gradients;
ІКТ	-	інформаційно-комунікаційні технології;
БПФ	-	перетворення Фур'є;
ІС	-	інтелектуальна система;
ІТ	-	інформаційні технології;
ОС	-	операційна система;
ПРО	-	проблема розпізнавання образів;
ПЗ	-	програмне забезпечення;
ПК	-	персональний комп'ютер;
ФВЧ	-	фільтр високих частот;
ФНЧ	-	фільтр низьких частот;
ШНМ	-	штучна нейронна мережа.

ВСТУП

Створення систем автоматичного розпізнавання графічних образів є актуальною проблемою, оскільки ідентифікація об'єктів, зафіксованих різними засобами спостереження, сприяє підвищенню безпеки життя та здоров'я людей.

Наприклад, під час пандемії влада запобігає поширення захворювання посиленням карантинних умов, що зокрема забороняє перебувати у громадському місці без захисної маски або респіратора. Впроваджуються штрафи за порушення карантину. Через «перебування у громадських будинках, спорудах, громадському транспорті під час дії карантину без респіратора чи захисної маски» можна отримати штраф від 170 до 255 грн [40]. Для спрощення виявлення порушників карантину (не носіння або неправильне носіння масок) у багатолюдних місцях, там, де встановлено камери спостереження, можливо автоматичне визначення таких порушників за допомогою програми-детектора.

Підходи до створення застосунку для розпізнавання графічних образів можна дослідити на прикладі розробки детектора маски.

Мета роботи – дослідження та розробка програмного застосунку для фіксації та розпізнавання відеоспостереження з використанням сучасних засобів розробки (Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask).

Для досягнення мети необхідно вирішити наступні **завдання**:

- аналіз методів розпізнавання графічних образів;
- особливості алгоритмічних побудов за класифікацією графічних образів;
- аналіз існуючих підходів до розробки системи розпізнавання графічних образів;
- розробка детектора маски для обличчя засобами Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask.

Об'єкт дослідження – методи розпізнавання графічних образів.

Предмет дослідження – технології створення додатку "Детектор маски", нейронної мережі.

Методи дослідження. Розробка, проектування і тестування нейронної мережі для полегшення процедури розпізнавання графічних образів.

Практичне значення полягає у можливості застосування отриманих результатів роботи для задачі розпізнавання образів.

Особистий внесок: розроблено визначник маски на обличчі засобами Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask..

Структура і обсяг роботи

Робота складається з вступу, трьох розділів, висновків списку використаних джерел, додатків. Обсяг роботи становить 71 сторінка, обсяг використаних джерел – 39 джерел.

Перший розділ містить аналіз методів розпізнавання графічних образів та особливостей існуючих методів. Надано огляд штучної нейронної мережі, як інструменту для вирішення задачі.

У другому розділі проводиться дослідження проблеми класифікації графічних образів, наведено алгоритмічні побудови та рецепторну структуру сприйняття інформації.

У третьому розділі описано роботу загальних модулів комп'ютерної системи розробленого вебзастосування "Детектора маски" для обличчя. Додатки містять головні елементи коду.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Завдання розпізнавання графічних образів тривалий час розглядалося переважно з біологічної та психологічної точок зору. У цих дослідженнях аналізувалися переважно якісні характеристики, які не давали змоги точно пояснити механізм функціонування. Зазвичай отримання функціональних залежностей пов'язували з дослідженням рецепторів органів чуття (слуху, дотику, зору), однак принципи ухвалення рішень залишалися невідомими. На ранніх етапах досліджень основною помилкою було уявлення про те, що мозок працює за чітко визначеними алгоритмами, і ці алгоритми можна відтворити за допомогою технічних засобів.

На початку XX століття Норберт Вінер заснував нову науку — кібернетику, що вивчає загальні закономірності управління і передачі інформації в машинах, живих організмах і суспільстві. Це дало змогу впровадити кількісні методи у вивчення розпізнавання образів, зокрема математичний опис цього процесу. Розробка пристроїв для розпізнавання об'єктів дозволила автоматизувати певні завдання, замінюючи людину спеціалізованими системами. Такі пристрої забезпечують стабільну якість роботи, оскільки не залежать від людського фактора (кваліфікації, досвіду, настрою тощо). Автоматизація також сприяє підвищенню ефективності складних систем через моніторинг, своєчасне обслуговування, виявлення перешкод і зниження шуму.

Проблема розпізнавання графічних образів давно привертає увагу фахівців із прикладної математики, а згодом і інформатики. У 1920-х роках Р. Фішер заклав основи дискримінантного аналізу, одного з методів теорії розпізнавання. У 1940-х А. Н. Колмогоров і А. Я. Хінчин досліджували розподіл суміші двох статистичних розподілів [37]. У 1950–1960-х роках сформувалася теорія статистичних рішень, що дала змогу розробляти алгоритми для класифікації об'єктів. Це стало початком планомірного

наукового пошуку і створення систем розпізнавання об'єктів, явищ і процесів. Нова дисципліна отримала назву "розпізнавання образів" [29].

Отже, класична теорія статистичних рішень стала основою для створення алгоритмів, які на основі експериментальних даних та апріорної інформації визначають клас, до якого належить об'єкт.

Розпізнавання образів (об'єктів) — це процес визначення або ідентифікації об'єкта, а також його властивостей на основі зображення (оптичне розпізнавання) або аудіозапису (акустичне розпізнавання) та інших характеристик (рис. 1.1) [34].

Образом називають класифікаційну категорію, що об'єднує певну групу об'єктів за спільною ознакою. Характерні властивості образів дозволяють, ознайомившись із обмеженим числом представників множини, розпізнавати нові елементи цієї ж множини.

Метод визначення, до якого образу належить елемент, називається вирішальним правилом. Іншим важливим поняттям є метрика — спосіб вимірювання відстані між елементами універсальної множини. Чим менша відстань, тим більш схожі об'єкти (наприклад, символи, звуки тощо). Зазвичай об'єкти представляють у вигляді наборів чисел, а метрику — у вигляді функції. Вибір способу представлення образів і типу метрики впливає на ефективність алгоритму: один і той самий метод із різними метриками може демонструвати різну точність розпізнавання.

Адаптація — це процес налаштування параметрів і структури системи або її керуючих впливів, що базується на поточній інформації, з метою досягнення заданого стану системи за умов невизначеності або змінних робочих умов.

Навчання — це процес, у результаті якого система поступово вчиться відповідати на певні зовнішні впливи потрібними реакціями. Адаптація, у свою чергу, забезпечує налаштування параметрів і структури системи для досягнення необхідної якості функціонування в умовах постійних змін зовнішнього середовища.

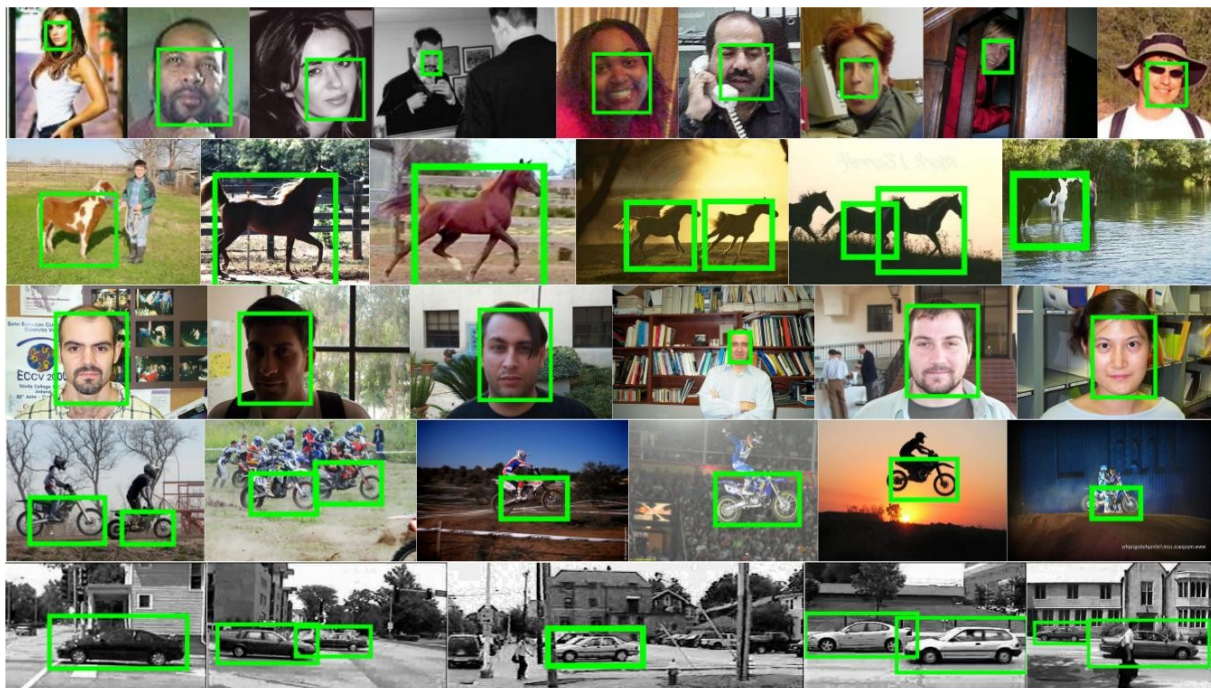


Рис. 1.1 – Приклад розпізнавання графічних образів

Під навчанням зазвичай розуміють вироблення реакцій системи на ідентичні групи зовнішніх сигналів через багаторазовий вплив і корекцію. Такі впливи можуть бути представлені у вигляді "заохочень" або "покарань", які формують механізм навчання.

Самонавчання відрізняється тим, що система не отримує додаткової інформації щодо правильності своїх реакцій [29].

1.1 Методи розпізнавання графічних образів

Методи розпізнавання графічних образів можна розділити на три групи: попередня фільтрація і підготовка зображення, логічна обробка результатів фільтрації, алгоритми прийняття рішень на основі логічної обробки. Межі між групами умовні. Для вирішення завдання далеко не завжди потрібно застосовувати методи з усіх груп, буває достатньо двох, а іноді навіть одного.

Фільтрація. До цієї групи відносяться методи, які дозволяють виділити на зображеннях області, без їх аналізу. Велика частина цих методів

застосовує єдине перетворення на всі точки зображення. На рівні фільтрації аналіз зображення не проводиться, але точки, які проходять фільтрацію, можна розглядати як області з особливими характеристиками.

Саме простіше перетворення - це бінаризація зображення по порогу. Для RGB зображення і зображення в градаціях сірого порогом виступає значення кольору. Існують ідеальні завдання, де цього перетворення достатньо. Наприклад, якщо потрібно автоматично виділити об'єкти на білому аркуші паперу, вибір порога для бінаризації значною мірою впливає на сам процес перетворення. У цьому випадку зображення було б бінаризоване на основі середнього значення кольору. Зазвичай для бінаризації використовують алгоритми, які адаптивно визначають оптимальний порі

Бінаризація може дати дуже цікаві результати при роботі з гістограмами, в тому числі в ситуації, якщо ми розглядаємо зображення не в RGB, а в HSV. Наприклад, сегментувати кольори. На цьому принципі можна побудувати як детектор мітки так і детектор шкіри людини.

Класична фільтрація: Фур'є, ФНЧ, ФВЧ. Класичні методи фільтрації з радіолокації і обробки сигналів можна з успіхом застосовувати в множині завдань Pattern Recognition. Традиційним методом в радіолокації, який майже не використовується в зображеннях в чистому вигляді, є перетворення Фур'є (БПФ). Одне з небагатьох винятків, при яких використовується одновимірне перетворення Фур'є, - компресія зображень. Для аналізу зображень одновимірного перетворення зазвичай не вистачає, потрібно використовувати куди більш ресурсоємне двовимірне перетворення.

$$G_{uw} = \frac{1}{NM} \sum_{n=1}^{N-1} \sum_{m=1}^{M-1} x_{mne^{-2\pi j \cdot}}$$

Мало хто його насправді розраховує, зазвичай, куди швидше і простіше використовувати згортку області з уже готовим фільтром, заточеним на високі (ФВЧ) або низькі (ФНЧ) частоти. Такий метод,

звичайно, не дозволяє зробити аналіз спектра, але в конкретному завданні відеоспостереження зазвичай потрібен не аналіз, а результат [2].

Найпростіші приклади фільтрів, що реалізують підкреслення низьких частот (фільтр Калмана (рис. 1.2, а)) і високих частот (Фільтр Габора, б).

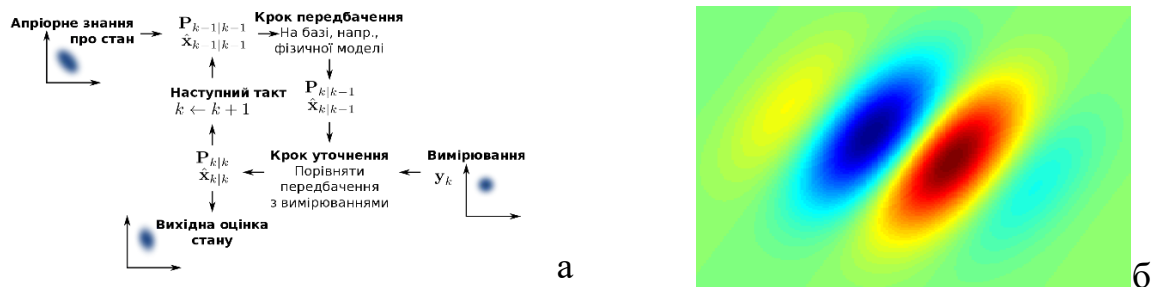


Рис. 1.2 – Приклади фільтрів

Для кожної точки зображення вибирається вікно і перемножується з фільтром того ж розміру. Результатом такої згортки є нове значення точки. При реалізації ФНЧ і ФВЧ виходять зображення такого типу (рис.1.2).

Але що якщо використовувати для згортки з сигналом якусь довільну характеристичну функцію, тоді це буде називатися "Вейвлет- перетворення". Це визначення не є коректним, але традиційно склалося, що в багатьох командах вейвлет-аналізом називається пошук довільного патерну на зображенні за допомогою згортки з моделлю цього патерну. Існує набір класичних функцій, використовуваних в вейвлет-аналізі. До них відносяться 3х-мірний вейвлет Хаара, 2х-мірний вейвлет Морлі, вейвлет мексиканський капелюх, вейвлет Добеши (рис. 1.3) [30].

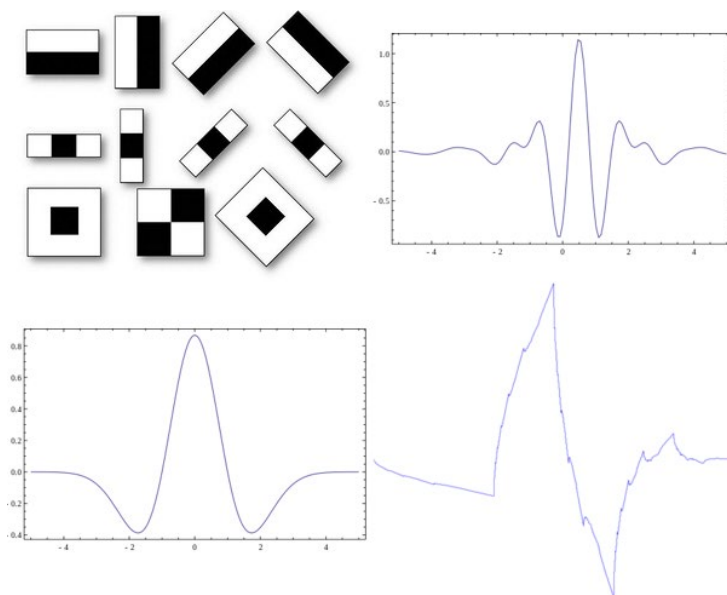


Рис. 1.3 – Приклади вейвлетів

Хорошим прикладом використання розширеної трактування вейвлетів є завдання пошуку відблиску в оці, для якого вейвлетом є сам відблиск (рис. 1.4).

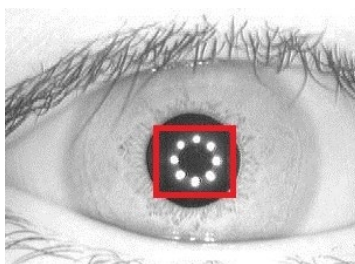


Рис. 1.4 – Приклад використання вейвлетів

Класичні вейвлети зазвичай використовуються для стиснення зображень, або для їх класифікації.

Після вільного трактування вейвлетів, варто згадати про кореляцію, що лежить в їх основі. При фільтрації зображень це незамінний інструмент. Класичне застосування - кореляція відеопотоку для знаходження зрушень або оптичних потоків. Найпростіший детектор зсуву - теж є різницеvim корелятором. Там де зображення не корелюють спостерігається рух.

Цікавим класом фільтрів є фільтрація функцій. Це чисто математичні фільтри, які дозволяють виявити просту математичну функцію на

зображенні (пряму, параболу, коло). Будується зображення, в якому для кожної точки вихідного зображення промальовується множина функцій, які її породжують. Найбільш класичним перетворенням є перетворення Хафа для прямих (рис. 1.5). У цьому перетворенні для кожної точки $(x; y)$ вимальовується множина точок $(a; b)$ пряма $y = ax + b$, для якої вірна рівність.

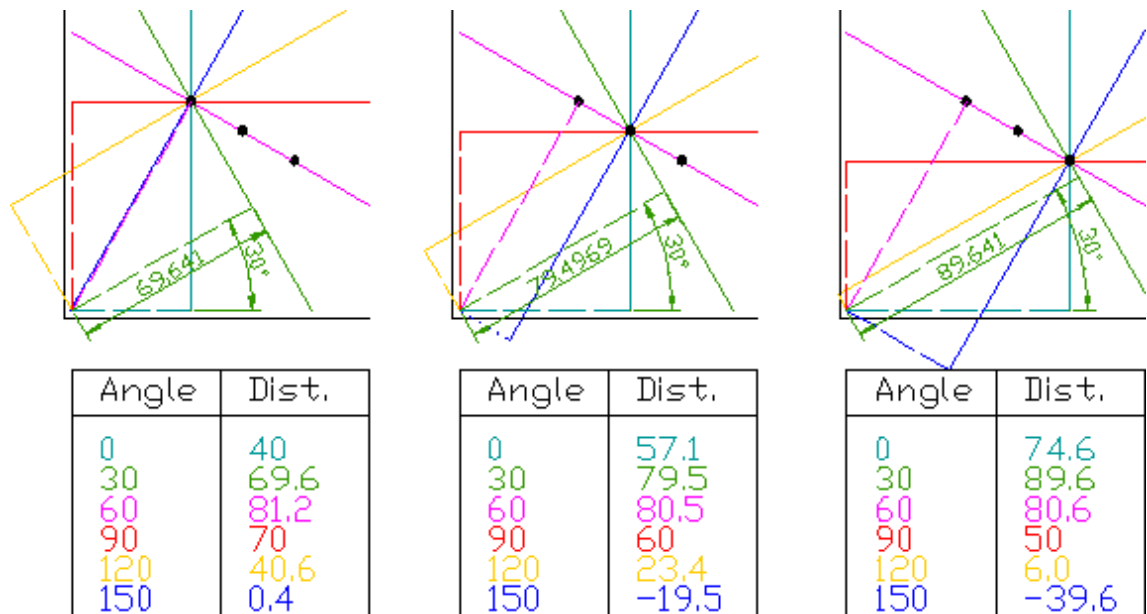


Рис. 1.5 – Перетворення Хафа для прямих

Перетворення Хафа дозволяє знаходити будь-які параметризовані функції. Наприклад кола. Існує модифіковане перетворення, яке дозволяє знаходити різноманітні фігури. Це перетворення широко застосовується в математиці, однак при обробці зображень його ефективність часто обмежена. Основними недоліками є низька швидкість роботи та висока чутливість до якості бінаризації. Навіть у сприятливих умовах зазвичай рекомендується використовувати альтернативні методи [3].

Аналогом перетворення Хафа для лінійних функцій є перетворення Радона, яке обчислюється за допомогою швидкого перетворення Фур'є (БПФ). Це забезпечує значний приріст продуктивності в ситуаціях, коли кількість точок дуже велика (рис.1.6) [13].

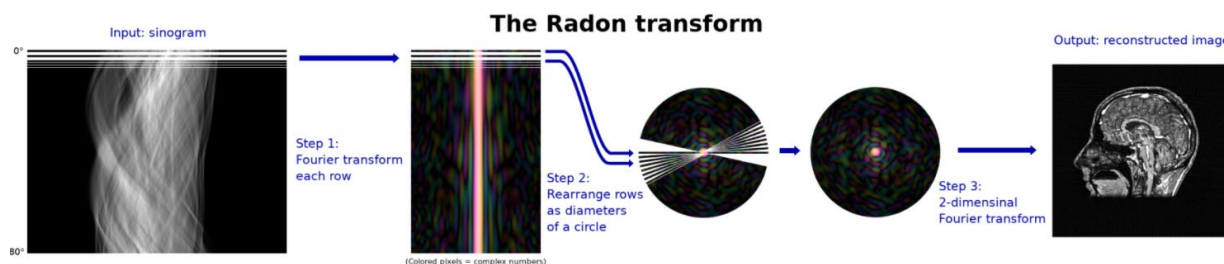


Рис. 1.6 – Обчислення двовимірного перетворення Радона через два перетворення Фур'є

Окремий клас фільтрів становить фільтрація меж і контурів. Контури відіграють важливу роль, коли необхідно перейти від обробки зображення до аналізу об'єктів на ньому. У випадках, коли об'єкт має складну форму, але чітко виділяється, виділення його контурів часто є єдиним ефективним методом роботи з ним. Існує багато алгоритмів, які виконують фільтрацію контурів, зокрема, алгоритм Canny, який є одним із найпоширеніших і часто використовується в бібліотеці комп'ютерного зору OpenCV.

Результати фільтрації надають набір даних, придатних для подальшої обробки. У багатьох випадках ці дані можна безпосередньо використовувати без додаткової обробки.

Переходом від фільтрації до логіки, є методи математичної морфології. По суті, це найпростіші операції нарощування і ерозії бінарних зображень. Ці методи дозволяють прибрати шуми з бінарного зображення, збільшивши або зменшивши наявні елементи. На базі математичної морфології існують алгоритми оконтурювання, але зазвичай користуються якимись гібридними алгоритмами або алгоритмами в зв'язці.

Алгоритми отримання границь. Отримання границі досить просто перетворюються в контури. Для оператора Canny це відбувається автоматично, для інших алгоритмів потрібна додаткова бінаризація [12].

Контур є унікальною характеристикою об'єкту. Часто це дозволяє ідентифікувати об'єкт по контуру. Існує потужний математичний апарат, що дозволяє це зробити, який називається контурним аналізом. Математичний апарат, що використовується для виділення особливих точок, називається

контурним аналізом. Особливі точки є унікальними характеристиками об'єкта, які дозволяють зіставляти його як із самим собою, так і з об'єктами зі схожих класів. Існує безліч способів виділення таких точок. Одні методи забезпечують визначення особливих точок у сусідніх кадрах, інші – за тривалих інтервалів часу або змін освітлення, а треті – дозволяють знаходити особливі точки, стійкі до поворотів об'єкта.

Розглянемо кілька класів методів, упорядкованих за зростанням складності:

Перший клас. Особливі точки, стабільні протягом короткого часу (секунд). Вони використовуються для відстеження об'єкта між сусідніми кадрами відео або для поєднання зображень із різних камер. До таких точок належать локальні максимуми, кути на зображенні (рис. 1.7), точки з максимальною дисперсією, певні градієнти тощо.

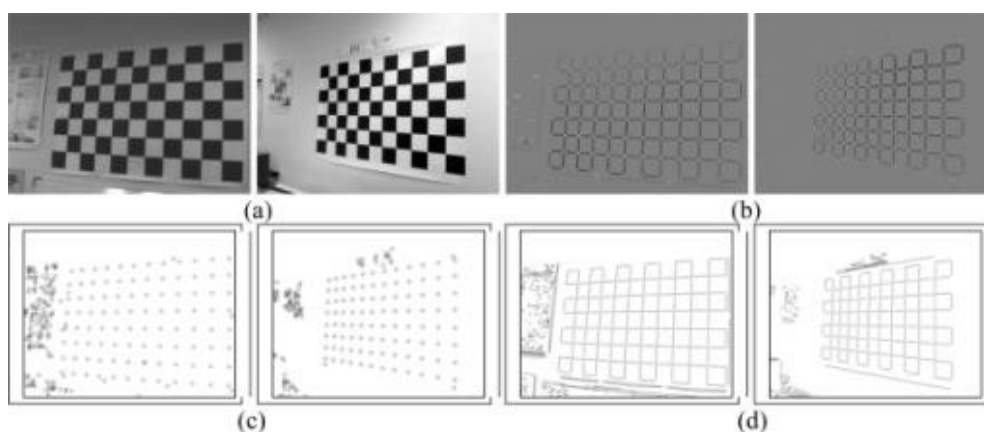


Рис. 1.7 – Розпізнавання кутів на зображеннях

Другий клас. Особливі точки, стійкі до змін освітлення та незначних переміщень об'єкта. Вони застосовуються для навчання моделей і класифікації об'єктів, таких як пішоходи чи обличчя. Основою для таких точок можуть бути певні вейвлети, наприклад, примітиви Хаара, відблиски, або інші специфічні функції. До цього класу також належать точки, визначені методом гістограм спрямованих градієнтів (HOG) (рис. 1.8).

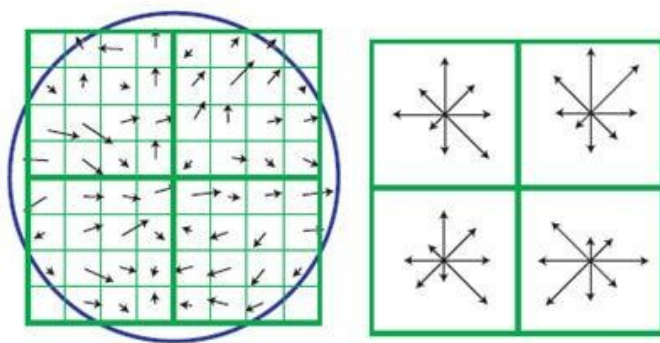


Рис. 1.8 – Гістограм спрямованих градієнтів [33]

Третій клас. Стабільні особливі точки, які залишаються незмінними навіть при поворотах об'єкта. До цього класу належать методи SURF і SIFT, що забезпечують високу стабільність. Розрахунок таких точок є тривалішим, ніж у попередніх методів, але все ще виконується за прийнятний час [19]. Ці методи, на жаль, запатентовані.

Алгоритм навчання. Розглянемо методи, які не працюють безпосередньо з зображенням, але які дозволяють приймати рішення. В загалі, це різні методи машинного навчання і прийняття рішень. Нехай це буде наявність / відсутність людини на фотографії.

Кожне зображення характеризується набором ознак, які були отримані за допомогою певного методу, такого як ознаки Хаара, HOG, SURF або інші вейвлети. Завдання алгоритму навчання полягає у створенні моделі, здатної аналізувати нові зображення і визначати, який об'єкт на них зображений [25].

Кожне з тестових зображень - це точка в просторі ознак. Її координати це вага кожного з ознак на зображенні. Нехай нашими ознаками будуть: «Наявність очей», «Наявність носа», «Наявність двох рук», «Наявність вух», та ін.. Всі ці ознаки ми виділимо існуючими у нас детекторами, які навчені на частини тіла, схожі на людські. Для людини в такому просторі буде коректною точка $[1; 1; 1; 1; \dots]$. Для мавпи точка $[1; 0; 1; 0; \dots]$ для коня $[1; 0; 0; 0; \dots]$. Класифікатор навчається за вибіркою прикладів. Але не на всіх фотографіях виділилися руки, на інших немає очей, а на третій у мавпи через помилку класифікатора з'явився людський ніс. Той, якого

навчають класифікатор людини автоматично розбиває простір ознак таким чином, щоб сказати: якщо перша ознака лежить в діапазоні $0.5 < x < 1$, другий $0.7 < y < 1, \dots$, тоді це людина [12].

По суті мета класифікатора – відмалювати в просторі ознаки області, характеристичні для об'єктів класифікації.

1.2 Штучні нейронні мережі

Дослідження штучних нейронних мереж пов'язано з тим, що вони дозволяють наблизитися до можливостей обробки інформації людським мозком, який являє собою надзвичайно складний, нелінійний, паралельний комп'ютер (систему обробки інформації). Мозок має здатність організовувати свої структурні компоненти, так звані нейрони, так, щоб вони могли виконувати конкретні задачі (такі як розпізнавання образів, обробку сигналів органів почуттів, моторні функції) в багато разів швидше, ніж можуть дозволити най швидкодійні сучасні комп'ютери. На сьогоднішній день існує багато прикладів використання штучних нейронних мереж для прогнозів, класифікації, оптимізації, розпізнавання образів та багато інших.

Нейронні мережі – обчислювальні структури, які моделюють прості біологічні процеси, що асоціюються з процесами людського мозку. Вони представляють собою системи, здатні до навчання шляхом аналізу позитивних і негативних впливів. Елементарним перетворювачем в даних мережах є штучний нейрон або просто нейрон (на рис. 1.9, а, представлено одношарову нейронну мережу), названий так за аналогією з біологічним прототипом (рис. 1.9, б) [39].

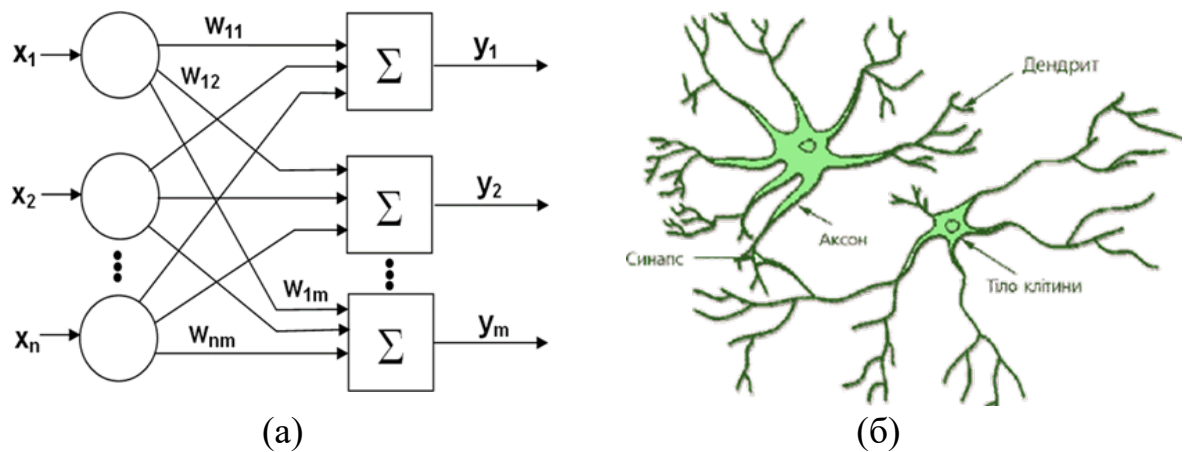


Рис. 1.9 – Штучний та біологічний нейрони

Прототипом для створення нейрона став біологічний нейрон головного мозку (рис. 1.9., б). Біологічний нейрон має тіло, сукупність відростків – дендритів, за якими нейрон надходять вхідні сигнали, і аксонів, що передають вихідні сигнали нейронів іншим клітинам. Точка з'єднання дендрита і аксона називається синапсом. Спрощено функціонування нейрона можна представити наступним чином:

1. Нейрон отримує від дендритів набір (вектор) вхідних сигналів.
2. У тілі нейрона оцінюється сумарне значення вхідних сигналів. Однак входи нейрона нерівнозначні. Кожен вхід характеризується деяким ваговим коефіцієнтом, що визначає важливість інформації переданої ним. Таким чином, нейрон не просто підсумовує значення вхідних сигналів, а обчислює скалярний добуток вектора вхідних сигналів і вектора вагових коефіцієнтів.
3. Нейрон формує вихідний сигнал, інтенсивність якого залежить від значення обчисленого скалярного перемноження. Якщо воно не перевищує деякого заданого порогу, то вихідний сигнал не формується зовсім – нейрон «не спрацьовує».
4. Вихідний сигнал надходить на аксон і передається дендриту інших нейронів [32].

1.2.1 Навчання нейронної мережі

Головною перевагою нейронних мереж є їхня здатність навчатися на основі даних із навколишнього середовища, що дозволяє з часом підвищувати продуктивність. Це вдосконалення відбувається відповідно до визначених правил. Навчання нейронної мережі здійснюється через інтерактивне коригування синаптичних ваг і порогових значень. У ідеальних умовах мережа здобуває нові знання про середовище з кожною ітерацією процесу навчання [39].

Процес навчання включає безліч різних підходів, тому його складно визначити однозначно. Більше того, уявлення про навчання змінюється залежно від точки зору. Наприклад, психолог і шкільний учитель дивляться на навчання по-різному. У контексті нейронних мереж навчання можна визначити як процес налаштування вільних параметрів мережі через моделювання середовища, в якому вона функціонує.

Тип навчання визначається методом підстроювання цих параметрів і включає такі етапи:

1. Отримання стимулів із зовнішнього середовища.
2. Зміна вільних параметрів нейронної мережі на основі цих стимулів, що призводить до іншої реакції на подібні впливи в майбутньому.

Послідовність дій для реалізації навчання мережі називається алгоритмом навчання. Універсального алгоритму, що підходив би для всіх архітектур, не існує. Замість цього є набір алгоритмів, кожен із яких має свої переваги та обмеження.

Алгоритми навчання розрізняються за підходом до налаштування синаптичних ваг і способом взаємодії нейронної мережі із зовнішнім середовищем. У цьому контексті говорять про парадигму навчання, яка залежить від моделі середовища, в якому функціонує мережа. Виділяють три основні типи навчання: навчання з учителем, навчання без учителя, змішане навчання.

Навчання нейронної мережі з учителем передбачає, що для кожного вхідного вектора із навчальної вибірки задане відповідне цільове значення вихідного вектора. Такі пари вхідних і цільових значень утворюють навчальну множину. У процесі навчання ваги мережі коригуються доти, доки для кожного вхідного вектора відхилення вихідного результату від цільового не стане прийнятно малим.

Нейронна мережа має у своєму розпорядженні правильними відповідями (виходами мережі) на кожен вхідний приклад. Ваги налаштовуються так, щоб мережа виробляла відповіді, як можна більш близькі до відомих правильних відповідей. Посилений варіант навчання з учителем припускає, що відома тільки критична оцінка правильності виходу нейронної мережі, але не самі правильні значення виходу.

Навчання нейронної мережі без учителя є більш природною моделлю з точки зору біологічних витоків штучних нейронних мереж. У цьому підході навчальна множина містить лише вхідні вектори. Алгоритм налаштовує ваги мережі так, щоб вихідні вектори були узгодженими, тобто схожі вхідні вектори давали однакові результати. Такий тип навчання не потребує знання правильних відповідей для кожного прикладу у вибірці. Він дозволяє виявляти внутрішню структуру даних або кореляції між зразками, що дає можливість групувати дані за категоріями. У разі змішаного навчання частина ваг встановлюється за допомогою методу з учителем, а решта – завдяки самонавчанню [37].

Основні застосування нейронних мереж:

Класифікація образів. Завдання полягає у визначенні приналежності вхідного образу (наприклад, мовного сигналу або рукописного символу), представленого у вигляді вектора ознак, до одного або кількох заздалегідь визначених класів. Приклади включають розпізнавання букв, мовлення, класифікацію електрокардіограм або клітин крові.

Кластеризація (категоризація). У задачах кластеризації, також відомих як класифікація образів без учителя, немає навчальної вибірки з мітками

класів. Алгоритм кластеризації групує схожі образи в один кластер, ґрунтуючись на їхній схожості. Це застосовується для стиснення даних і аналізу їхніх властивостей.

Апроксимація функцій. Уявімо, що навчальна вибірка містить пари вхідних і вихідних даних, згенеровані невідомою функцією $F(x)$, спотвореною шумом. Мета апроксимації – знайти оцінку цієї функції. Такі задачі часто виникають у науковому і технічному моделюванні.

Прогнозування. Це завдання полягає у передбаченні значень у майбутні моменти часу. Прогнозування широко застосовується в бізнесі, науці та техніці, наприклад, для передбачення цін на фондовому ринку чи прогнозу погоди.

Оптимізація. Багато задач у математиці, статистиці, техніці, медицині та економіці формуються як задачі оптимізації. Мета – знайти рішення, яке відповідає заданим обмеженням і максимізує або мінімізує цільову функцію. Наприклад, задача комівояжера є класичним прикладом оптимізації.

Пам'ять, що адресується за змістом. У традиційній обчислювальній моделі фон Неймана доступ до пам'яті здійснюється через її адреси, незалежно від її вмісту. У разі помилки адресації можна отримати невірну інформацію. Асоціативна пам'ять або пам'ять, що адресується за змістом, працює по-іншому: доступ здійснюється на основі вказаного вмісту. Така пам'ять може знаходити дані навіть за частковим запитом, що робить її ефективною для мультимедійних баз даних.

Адаптація до змін навколишнього середовища. Нейронні мережі мають здатність адаптуватися до змін навколишнього середовища. Зокрема, нейронні мережі, навчені діяти в певному середовищі, можуть бути легко перевчені для роботи в умовах незначних коливань параметрів середовища. Більш того, для роботи в нестаціонарній середовищі (де статистика змінюється з плином часу, наприклад як котирування акцій на біржі) можуть бути створені нейронні мережі, вони перенавчаються в реальному часі. Чим вище адаптивні здатності системи, тим більш стійкою буде її робота в

нестационарної середовищі. При цьому слід зауважити, що адаптивність не завжди веде до стійкості; іноді вона призводить до зовсім протилежного результату. Наприклад, адаптивна система з параметрами, що швидко змінюються в часі, може також швидко реагувати і на сторонні порушення, що викличе втрату продуктивності. Для того, щоб використовувати всі переваги адаптивності, основні параметри системи повинні бути досить стабільними, щоб можна було не враховувати зовнішні перешкоди, і досить гнучкими, щоб забезпечити реакцію на істотні зміни середовища.

Висока швидкість проведення обчислювань. Нейронні мережі мають потенційну надвисоку швидкодію за рахунок використання масового паралелізму обробки інформації.

Нейронні мережі потенційно відмовостійкі при апаратній реалізації. Це означає, що за несприятливих умов їх продуктивність падає незначно. Наприклад, якщо пошкоджений якийсь нейрон або його зв'язки, вилучення запам'ятованої інформації ускладнюється. Проте, враховуючи розподілений характер зберігання інформації в нейронній мережі, можна стверджувати, що тільки серйозні пошкодження структури нейронної мережі суттєво вплинуть на її працездатність. Тому нейронні мережі чудово підходять для розпізнавання образів на зображеннях [6, 24, 37].

1.3 Висновки до розділу 1

На основі аналізу методів виявлення та розпізнавання обличчя, порівняння їх можливостей, обґрунтовано вибір методів в якості алгоритмів в розроблюваній системі. Для реалізації проєкту обрано методи локальних бінарних шаблонів в якості алгоритму розпізнавання обличчя, що має бути покладений в основу розроблюваної системи; алгоритм навчання; оператор Canny, який найчастіше використовується в бібліотеці комп'ютерного зору OpenCV. Проаналізовано роботу штучного та біологічного нейронів, виявлено основні проблеми, які можуть виникнути в ході вирішення поставленого завдання.

РОЗДІЛ 2

ПРОБЛЕМИ КЛАСИФІКАЦІЇ ГРАФІЧНИХ ОБРАЗІВ

2.1 Рецепторна структура сприйняття інформації

Для того, щоб людина свідомо сприймала інформацію, вона повинна пройти досить тривалий цикл попередньої обробки. Спочатку світло потрапляє в око. Пройшовши через всю оптичну систему фотони, врешті-решт, потрапляють на сітківку - шар світлочутливих клітин - паличок і колбочок.

Тепер інформація надходить по зоровому нерву в головний мозок людини, в так звані "зорові горби", на які проектується зорова інформація, те, що ми бачимо. Далі зорова інформація надходить до відділів мозку, які вже виділяють з неї окремі складові - горизонтальні, вертикальні, діагональні лінії, контури, області світлого, темного, кольорового. Поступово образи стають все більш складними і розмитими, але графічний образ картини пройде ще довгий шлях, перш ніж досягне рівня свідомості.

До проблеми розпізнавання образів в інформатиці можна підходити, відштовхуючись від аналогії з біологічними процесами. У деяких умовах здатності тварин до розпізнавання образів перевищують здатності будь-якої машини, яку тільки можна побудувати.

При класифікації, заснованій на безпосередньому сенсорному досвіді, тобто при розпізнаванні осіб або вимовлених слів, люди легко перевершують технічні пристрої. В "несенсорним ситуаціях" дії людей не настільки ефективні. Наприклад, люди не можуть змагатися з програмами класифікації образів, якщо правильний спосіб класифікації включає логічні комбінації абстрактних властивостей, таких, як колір, розмір і форма. Оскільки розпізнавання образів має бути функцією нейронів тварин, можна шукати ключ до біологічного розпізнавання образів у властивостях самого нейрона. Для багатьох цілей нейрон можна розглядати як граничний елемент. Це означає, що він або дає на виході деяку постійну величину, якщо сума його

входів досягає певного значення, або ж залишається пасивним. Мак-Каллок і Піттс довели, що будь-яку обчислювану функцію можна реалізувати за допомогою належним чином організованої мережі ідеальних нейронів - порогових елементів, логічні властивості яких можна приписати реальному нейрону. Проблема полягає в тому, чи можна знайти розумний принцип реорганізації мережі, що дозволяє випадково об'єднаній групі ідеальних нейронів самоорганізуватися в "обчислювальний пристрій", здатний вирішувати довільну задачу розпізнавання образів. Такий принцип реорганізації був позначений як теорія навчання.

Нейрологічна теорія навчання, висунута канадським психологом Хеббом, була розрахована на використання в якості моделі, призначеної для психології, справила великий вплив на штучний інтелект. Її модифікація застосовувалася при визначенні принципів системи розпізнавання образів, що одержали назву персептрон. Персептрони, описані Розенблаттом, можуть існувати і в формі програм, і як спеціально сконструйовані обчислювальні машини.

Образ, або клас, — це категорія в системі класифікації, яка об'єднує групу об'єктів на основі спільної ознаки. Здатність до образного сприйняття світу є однією з властивостей живого мозку, яка допомагає впорядковувати безмежний потік інформації та зберігати орієнтацію серед розрізнених даних про навколишній світ.

Сприймаючи зовнішній світ, завжди проводиться класифікація інформації. Розбивка її на групи схожих, але не тотожних явищ. Наприклад, незважаючи на істотну відмінність, до однієї групи належать всі букви "А", написані різними почерками, або все звуки, відповідні одній і тій самій ноті, взятої в будь-якій октаві і на будь-якому інструменті. Для складання поняття про групу сприйняття досить ознайомитися з незначною кількістю її представників. Ця властивість мозку дозволяє сформулювати таке поняття, як образ.

Образи мають характерну властивість, що виявляється в тому, що ознайомлення з певним числом явищ однієї групи дає можливість визначати будь-яких її представників.

Різні люди, навчаючись на різних наборах спостережень, зазвичай класифікують однакові об'єкти подібним чином і незалежно один від одного. Саме ця уніфікованість образів забезпечує взаєморозуміння між людьми в усьому світі.

Здатність сприймати навколишній світ у формі образів дає можливість із певною ймовірністю розпізнавати необмежену кількість об'єктів на основі ознайомлення лише з їхньою обмеженою кількістю. Об'єктивність основних властивостей образів дозволяє моделювати процес їх розпізнавання.

Розпізнавання образів полягає у визначенні об'єкта або його характеристик за допомогою його зображення (оптичне розпізнавання) чи аудіозапису (акустичне розпізнавання).

Внаслідок біологічної еволюції багато тварин вирішили це завдання за допомогою зорового і слухового апарату. Не зважаючи на це, створення штучних систем з функціями розпізнавання образів залишається складною технічною проблемою.

Загалом, проблема розпізнавання образів включає дві ключові складові: навчання та розпізнавання. Під час навчання системі демонструються окремі об'єкти із зазначенням їхньої належності до певної групи. У результаті система повинна навчитися однаково реагувати на всі об'єкти, що належать до одного образу, і по-іншому — на об'єкти, що належать до інших образів.

Важливо, що навчання здійснюється шляхом показу лише обмеженої кількості об'єктів. Навчальними об'єктами можуть бути зображення, наприклад, картинки (рис. 2.1), літери чи цифри [29]. При цьому в процесі навчання зазначаються лише самі об'єкти та їхня належність до певного образу.

Після етапу навчання відбувається розпізнавання нових об'єктів, що демонструє роботу вже навченої системи. Автоматизація цих процесів становить суть проблеми навчання розпізнаванню образів.

У тих випадках, коли людина самостійно створює або визначає правило класифікації та передає його машині, проблема розпізнавання вирішується лише частково, оскільки основний і найважливіший етап — навчання — виконує сама людина.



Рис. 2.1 – Приклад об'єктів навчання

Коло завдань, які можна вирішувати за допомогою систем розпізнавання, є дуже широким. Це включає не лише задачі розпізнавання візуальних та аудіальних образів, а й завдання класифікації складних процесів і явищ, таких як вибір стратегічних рішень керівником підприємства або визначення оптимального напрямку для управління технологічними, економічними, транспортними або військовими завданнями. Для аналізу будь-якого об'єкта необхідно спочатку отримати впорядковану інформацію про нього.

Вибір початкового опису об'єктів є однією з ключових проблем у задачах розпізнавання образів. Якщо початковий опис обраний правильно, задача розпізнавання може бути розв'язана досить легко, тоді як невдалий вибір може призвести до складних труднощів у подальшій обробці інформації або навіть до відсутності рішення.

Будь-яке зображення, отримане в результаті спостереження об'єкта під час навчання або тестування, можна представити у вигляді вектора, що відповідає точці в певному просторі ознак. Якщо можна однозначно віднести зображення до одного з кількох образів, це означає, що в просторі існують області, які не перетинаються, і зображення належить одній з цих областей. Кожній такій області можна присвоїти назву, яка буде відповідати певному образу.

Для кращого розуміння процесу адаптації розпізнавання образів можна застосувати аналогії з геометрією, розглядаючи випадок розпізнавання лише двох образів. Відомо тільки те, що необхідно розділити дві області в деякому просторі, але точні межі цих областей і правила визначення їх приналежності залишаються невідомими.

Під час навчання подаються точки, випадковим чином вибрані з різних областей, і надається інформація про те, до якої області належать ці точки. Не дається ніякої додаткової інформації про ці області або їхні межі під час навчання. Основна мета навчання полягає в побудові поверхні, яка б не лише відокремлювала показані під час навчання точки, а й розділяла всі інші точки, що належать цим областям, або в побудові поверхонь, що обмежують ці області таким чином, щоб кожна з них містила лише точки одного образу. Завдання навчання полягає в створенні таких функцій, які, наприклад, даватимуть позитивні значення на всіх точках одного образу і негативні на всіх точках іншого.

Якщо зображення належать не двом, а кільком образам, завдання змінюється на побудову поверхні, яка б розділяла всі області, що відповідають цим образам. Це завдання можна вирішити, створивши функцію, яка дає

однакове значення для точок однієї області і різні значення для точок з інших областей.

На перший погляд може здатися, що для відокремлення всієї області достатньо лише кількох точок з цієї області. Дійсно, можна навести безліч різних областей, що містять ці точки, і як би не була побудована поверхня, що виділяє область, завжди знайдеться інша область, яка перетинає цю поверхню, але містить вказані точки. Проте відомо, що задача наближення функції на основі обмеженого набору точок є типовою математичною задачею апроксимації функцій. Для її вирішення необхідно ввести певні обмеження на клас розглянутих функцій, і вибір цих обмежень залежить від типу інформації, яку може надати вчитель під час навчання. Однією з таких підказок є гіпотеза про компактність образів.

Апроксимація функції розділу буде завданням легшим для більш компактних груп, аніж для тих, що більш рознесені в просторі області. Так, наприклад, в прикладі, показаному на рис. 2.2а, поділ свідомо більший ніж в прикладі, показаному на рис. 2.2б. Дійсно, в прикладі, зображеному на рис. 2.2а, області можуть бути розділені площиною, і навіть при великих погрішностях у визначенні функції вона буде продовжувати розділяти області. У разі ж на рис. 2.2б, поділ здійснюється нестандартною поверхнею, і навіть незначні відхилення в її формі призводять до помилок поділу. Саме це інтуїтивне уявлення про порівняно легку можливість поділу областей призвело до гіпотези компактності.

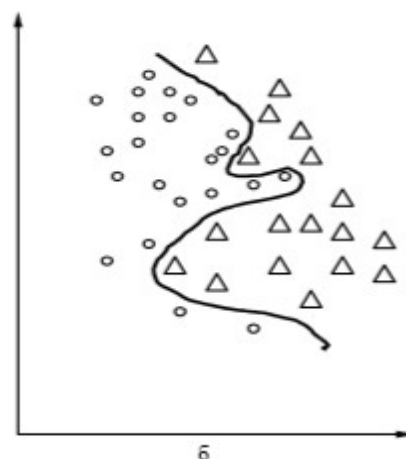
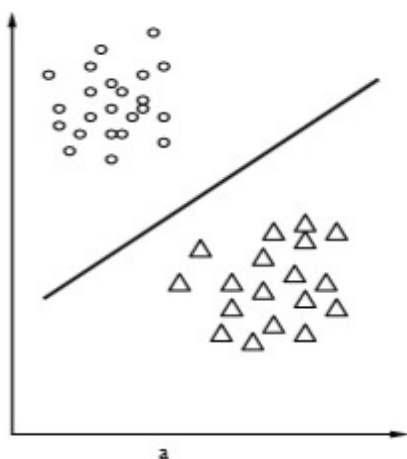


Рис. 2.2 – Розподіл двох образів у просторі

Окрім геометричної інтерпретації проблеми адаптації розпізнавання образів, існує також структурний (або лінгвістичний) підхід. Лінгвістичний підхід стає зрозумілим на прикладі розпізнавання візуальних зображень. Спочатку визначається набір базових понять — типових фрагментів зображень і характеристик їх взаємного розташування, таких як "зліва", "знизу", "всередині" тощо. Ці базові поняття створюють словник, що дозволяє формувати різні логічні висловлювання. Завдання полягає в тому, щоб з великої кількості можливих висловлювань, що можна створити за допомогою цих понять, вибрати найбільш важливі для конкретного випадку.

Наступним кроком є перегляд обмеженого набору об'єктів з кожного образу і побудова їх опису. Ці описи повинні бути достатньо детальними, щоб дозволити вирішити, до якого образу належить конкретний об'єкт. У процесі реалізації лінгвістичного підходу виникають дві основні задачі: створення початкового словника, який включає типові фрагменти, і розробка правил опису, що складаються з елементів цього словника.

У межах лінгвістичної інтерпретації проводиться аналогія між структурою зображень і синтаксисом мови. Ця аналогія стала можливою завдяки застосуванню інструментів математичної лінгвістики, оскільки методи є за своєю природою синтаксичними. Використовувати математичну лінгвістику для опису структури зображень можна лише після виконання сегментації зображень на складові частини, а також розробки "слів" для опису типових фрагментів і методів їх пошуку. Після виконання цієї підготовчої роботи виникають лінгвістичні завдання, що включають автоматичний граматичний розбір описів для розпізнавання зображень. Це створює самостійну область досліджень, що вимагає не тільки знання основ математичної лінгвістики, а й засвоєння технік, спеціально розроблених для лінгвістичної обробки зображень.

Якщо припустити, що в процесі навчання простір ознак формується на основі заданої класифікації, то задача в цьому просторі сама по собі задає властивість, за якою образи в ньому можна легко розділити. Гіпотеза компактності означатиме, що компактні образи відповідають компактним множинам в просторі ознак. Під компактними множинами маються на увазі "згустки" точок у просторі зображень, де між ними існують області розрідження. Цю гіпотезу не завжди вдавалося підтвердити експериментально, але в тих випадках, коли вона добре працювала (наприклад, на рисунку 2.2а), задача розпізнавання мала просте рішення. Навпаки, у тих випадках, коли гіпотеза не підтверджувалась (рис. 2.2б), задача або не вирішувалась, або вирішувалась з великими труднощами. Цей факт викликав сумніви щодо справедливості гіпотези компактності, оскільки для спростування будь-якої гіпотези достатньо одного прикладу, який її спростовує. Однак в тих випадках, коли гіпотеза виконувалась, що дозволяло успішно розв'язувати задачі навчання розпізнаванню образів, інтерес до цієї гіпотези залишався. Гіпотеза компактності перетворилась на критерій можливості задовільного вирішення задач розпізнавання.

Навчання — це процес, в якому система поступово набуває здатність давати потрібні реакції на певні сукупності зовнішніх впливів, тоді як адаптація — це налаштування параметрів і структури системи для досягнення необхідної якості управління в умовах змінюваних зовнішніх умов. Всі картинки на рисунку 2.1 ілюструють завдання навчання, в яких надаються кілька прикладів (навчальна послідовність) правильно вирішених задач. Якщо б вдалося виявити загальну властивість, що залежить не від природи образів чи їх зображень, а лише від їх здатності до поділу, то поряд з типовим завданням адаптації розпізнавання можна було б сформулювати задачу навчання без учителя. У такій задачі системі одночасно або послідовно надаються об'єкти без вказівок про їх приналежність до певних образів. Вхідний пристрій системи відображає множину об'єктів на множину зображень і, використовуючи закладену властивість поділу образів, здійснює

самостійну класифікацію. Після цього процесу самонавчання система повинна здобути здатність розпізнавати не тільки вже знайомі об'єкти (з навчальної послідовності), а й нові, яких раніше не було в навчанні. Процес самонавчання полягає в тому, що система без підказки вчителя набуває здатність давати однакові реакції на зображення об'єктів одного й того ж образу та різні реакції на зображення різних образів. Роль вчителя полягає лише в наданні системі певної об'єктивної властивості, спільної для всіх образів, що визначає здатність до поділу множин об'єктів на образи. Такою об'єктивною властивістю є компактність образів. Розташування точок в обраному просторі вже містить інформацію про те, як слід розділяти множини точок, і ця інформація є достатньою для самонавчання системи розпізнаванню образів.

Навчання зазвичай визначається як процес формування в системі певної реакції на групи однакових зовнішніх сигналів шляхом багаторазового впливу коригуючих зовнішніх факторів. Ці коригуючі фактори в контексті навчання часто називаються "заохоченнями" та "покараннями". Механізм, який генерує ці коригування, майже повністю визначає алгоритм навчання. Самонавчання ж відрізняється від традиційного навчання тим, що система не отримує зовнішніх вказівок щодо правильності своїх реакцій.

Більшість існуючих алгоритмів самонавчання здатні виокремлювати лише абстрактні образи — компактні множини в певних просторах. Відмінність між ними полягає в тому, як саме формалізується поняття компактності. Результати самонавчання свідчать про те, наскільки обраний простір підходить для конкретної задачі розпізнавання образів. Якщо абстрактні образи, які система виділяє під час самонавчання, збігаються з реальними, то простір вибрано вдало. Якщо ж ці образи значно відрізняються від реальних, то обраний простір не є оптимальним для вирішення конкретної задачі.

Адаптація — це процес зміни параметрів та структури системи, а також, можливо, коригування керуючих впливів на основі поточної інформації з

метою досягнення бажаного стану системи в умовах початкової невизначеності та змінюваних робочих умов.

Можливий спосіб побудови системи машинного розпізнавання, засновано на розрізненні будь-яких ознак, що виникають при розпізнаванні фігур. В якості ознак можуть бути обрані різні особливості фігур, наприклад, їх геометричні властивості (характеристики складових фігури кривих), топологічні властивості (взаємне розташування елементів фігури). Відомі системи розпізнавання, в яких розпізнання букв або цифр проводиться, за так званим "методом зондів" (рис. 2.3), тобто за кількістю перетинів контуру фігури з кількома особливим чином розташованими прямими.

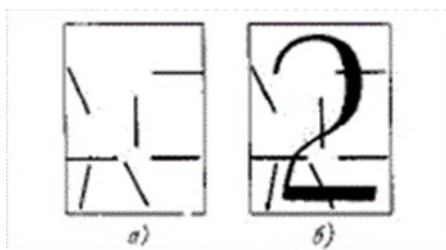


Рис. 2.3 – Схема розпізнавання цифр

Якщо проєктувати цифри на полі з зондами, то виявиться, що кожна з цифр перетинає цілком певні зонди, причому комбінації перетинів зондів різна у всіх десяти цифр. Ці комбінації і використовуються в якості ознак, за якими проводиться розрізнення цифр. Такі системи успішно справляються, наприклад, з читанням машинописного тексту, але їх можливості обмежені тим шрифтом (або групою подібних шрифтів), для якого була розроблена система ознак [35]. Робота зі створення набору еталонних фігур або системи ознак повинна проводитися людиною. Якість роботи системи і надійність "впізнавання" пропонованих фігур визначаються якістю попередньої підготовки і без участі людини не можуть бути підвищені.

Для того щоб ввести зображення в машину, потрібно перевести його в машинну мову, тобто закодувати, представити у вигляді деякої комбінації символів, якими може оперувати машина. Краще прагнути до найбільш "природного" кодування зображень. Як приклад, малювати фігури на деякому

полі, розбитому вертикальними і горизонтальними прямими на однакові елементи - квадратики. Елементи, на які попало зображення, суцільно фарбувати чорним, інші - залишати білими. Позначати чорні елементи одиницею, білі - нулем. Ввести послідовну нумерацію всіх елементів поля, наприклад, в кожному рядку зліва направо і по рядках зверху вниз. Тоді кожна фігура, намальована на такому полі, буде однозначно відображатися кодом, що складається з стількох цифр (одиниць і нулів), скільки елементів містить поле.

Таке кодування (рис. 2.4) вважається "природним" тому, що розбиття зображення на елементи лежить в основі роботи зорового апарату людини. Дійсно, сітківка ока складається з великого числа окремих чутливих елементів (так званих паличок і колбочок), пов'язаних нервовими волокнами із зоровими відділами головного мозку. Чутливі елементи сітківки передають за своїми нервових волокнах у головний мозок сигнали, інтенсивність яких залежить від освітленості даного елемента. Таким чином, зображення, спроектоване оптичною системою ока на сітківку, розбивається паличками і колбочками на окремі ділянки, і за основними елементами в деякому коді передається в мозок. Окремі елементи поля називаються рецепторами, а саме поле - полем рецепторів.

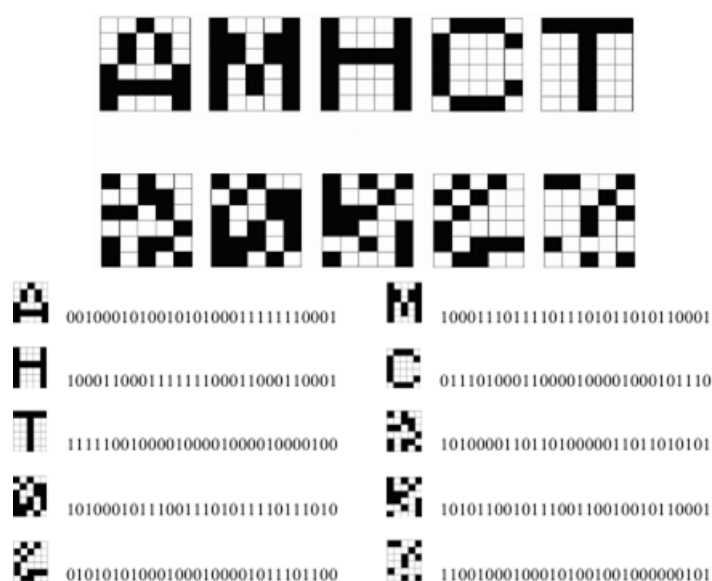


Рис. 2.4 – Приклади кодування зображень

Сукупність усіх плоских фігур, які можна зобразити на поле рецепторів, складає деяка множина. Кожна конкретна фігура з цієї сукупності є об'єктом цієї множини. Будь-якому з таких об'єктів відповідає певний код. Точно також будь-якому коду відповідає певне зображення на поле рецепторів. Взаємно однозначна відповідність між кодами і зображеннями дозволить оперувати тільки кодами, пам'ятаючи про те, що зображення завжди може бути відтворене за його кодом.

Ємність штучною нейронної мережі (ШНМ) - число образів, що пред'являються на вході ШНМ для розпізнавання. Для поділу множини вхідних образів, наприклад, за двома класами достатньо всього одного виходу. При цьому кожен логічний рівень - "1" і "0" - буде позначати окремий клас. На двох виходах можна закодувати вже 4 класу і так далі. Для підвищення достовірності класифікації бажано ввести надмірність шляхом виділення кожному класу одного нейрона у вихідному шарі або, що ще краще, декількох, кожен з яких навчається визначати приналежність образу до класу зі своїм ступенем достовірності, наприклад: високого, середнього та низької. Такі ШНМ дозволяють проводити класифікацію вхідних образів, об'єднаних в нечіткі (розмиті або пересічні) множини. Це властивість наближає подібні ШНМ до умов реального життя.

2.2 Алгоритмічні побудови

Алгоритмічна універсальність ЕОМ означає, що на них можна програмно реалізовувати (у вигляді машинної програми) будь-які алгоритми перетворення інформації, будь то обчислювальні алгоритми, алгоритми управління, пошуку доведення теорем, тривимірні графічні або аудіо композиції.

Однак, обчислювальні машини і роботи можуть в принципі вирішувати не всі завдання [21]. Було доведено існування таких типів завдань, для яких не можуть бути єдиного і ефективного алгоритму, що вирішує всі завдання даного типу; рішення таких задач за допомогою обчислювальних машин. Цей

факт сприяє кращому розумінню того, що можуть робити машини і чого вони не можуть зробити.

Серед властивостей штучних нейронних мереж основним є їх здатність до навчання, вони навчаються найрізноманітнішими методами. Більшість методів навчання походять від загальних передумов, і мають багато ідентичних характеристик. Їх навчання нагадує процес інтелектуального розвитку людської особистості. Можливості навчання штучних нейронних мереж обмежені. Мережа навчається, щоб для деяких множин входів давати необхідні множини виходів. Кожна така вхідна (або вихідна) множина розглядається як вектор. Навчання здійснюється шляхом послідовного пред'явлення вхідних векторів з одночасним підстроюванням ваг відповідно до певної процедури. У процесі навчання ваги мережі поступово стають такими, щоб кожен вхідний вектор виробляв вихідний вектор.

Навчальні алгоритми можуть бути класифіковані як алгоритми навчання з вчителем і без вчителя.

При навчанні з учителем існує вчитель, який пред'являє вхідні образи мережі, порівнює результуючі виходи з необхідними значеннями, а потім налаштовує ваги мережі таким чином, щоб зменшити відмінності. Навчання з учителем передбачає, що для кожного вхідного вектора існує цільовий вектор, що представляє собою необхідний вихід [21]. Разом вони називаються навчальною парою. Зазвичай мережа навчається на деякому числі таких навчальних пар. Пред'являється вихідний вектор, обчислюється вихід мережі і порівнюється з відповідним цільовим вектором, різниця (помилка) за допомогою зворотного зв'язку подається в мережу, і ваги змінюються відповідно до алгоритму, що прагне мінімізувати помилку. Вектори навчальної множини пред'являються послідовно, обчислюються помилки і ваги підлаштовуються для кожного вектора до тих, поки помилка по всьому навчальному масиву не досягне допустимого низького рівня.

Навчання з вчителем критикувалося за свою біологічну неправдоподібність. Важко уявити навчальний механізм в мозку, який би

порівнював бажані і дійсні значення виходів, виконуючи корекцію за допомогою зворотного зв'язку. Навчання без вчителя є набагато більш правдоподібною моделлю навчання в біологічній системі. Розвинена Кохоненом і іншими, вона не потребує цільові вектори для виходів і, отже, не вимагає порівняння з зумовленими ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто. щоб пред'явлення досить схожих вхідних векторів давало однакові виходи. Процес навчання, отже, виділяє статистичні властивості навчальної множини і групує подібні вектори в класи. Пред'явлення на вхід вектора з даного класу дасть певний вихідний вектор, але до навчання неможливо передбачити, який вихід буде проводитися даним класом вхідних векторів. Отже, виходи подібної мережі повинні трансформуватися в деяку зрозумілу форму, зумовлену процесом навчання.

Процес навчання ШНМ нового класу задач включає наступні стадії [37]:

1. Формулюється постановка задачі і виділяється набір ключових параметрів, що характеризують предметну область.
2. Вибирається парадигма нейронної мережі (модель, що включає в себе вид вхідних даних, порогової функції, структури мережі і алгоритмів навчання), найбільш підходяща для вирішення даного класу задач. Як правило, сучасні нейропакет, нейроплата й еволюційний [8] дозволяють реалізувати не одну, а кілька базових парадигм.
3. Готується, можливо, більш широкий набір навчальних прикладів, організованих у вигляді наборів вхідних даних, асоційованих з відомими вихідними значеннями. Вхідні значення для навчання можуть бути неповні і частково суперечливі.
4. Вхідні дані по черзі пред'являються ШНМ, а отримане вихідне значення порівнюється з еталоном. Потім проводиться підстроювання вагових коефіцієнтів міжнейронних з'єднань для мінімізації помилки між реальним і бажаним виходом мережі.

5. Навчання повторюється до тих пір, поки сумарна помилка у всій множини вхідних значень не досягне прийнятного рівня, або ШНМ не прийде в стаціонарний стан. Розглянутий метод навчання нейроподібні мережі носить назву «зворотне поширення помилки» (error backpropagation) і відноситься до числа класичних алгоритмів нейроматематики.

Налагоджена і навчена ШНМ може використовуватися на реальних вхідних даних, не тільки підказуючи користувачеві коректне рішення, а й оцінюючи ступінь його достовірності.

Існують різні алгоритми, що дозволяють розпізнавати образи. Алгоритм навчання машини "впізнавання" образів, заснований на методі січних гіперплощин (рис. 2.5), полягає в апроксимації розділу гіперповерхні "шматками" гіперплощин і складається з наступних основних етапів:

1. Навчання. Формування розділеної поверхні: проведення січних площин, вирізання зайвих площин, вирізання зайвих шматків площин.
2. Розпізнавання нових об'єктів.

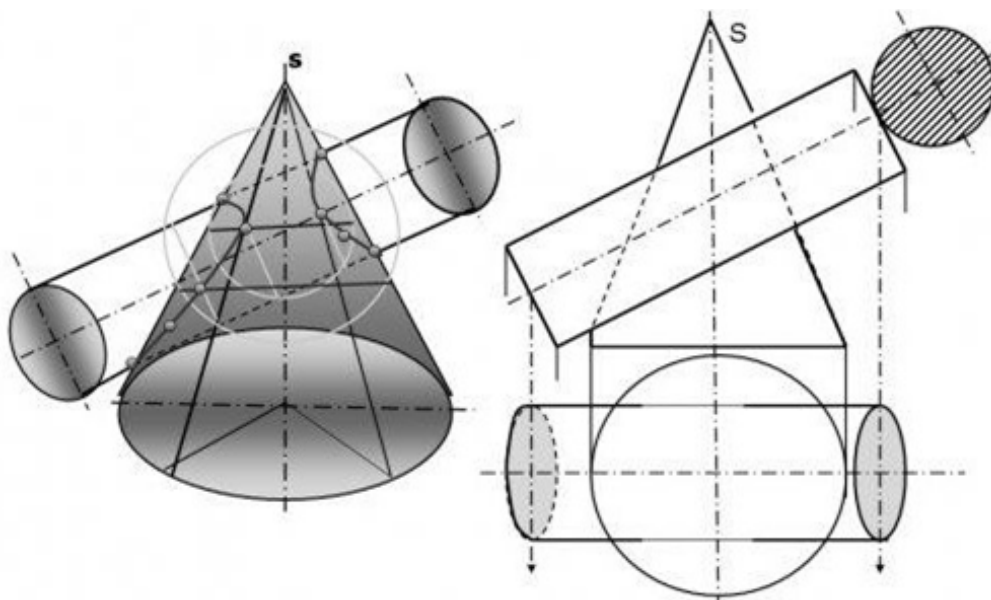


Рис. 2.5 – Метод січних гіперплощин

При використанні методу паралельних варіантів одночасно і незалежно один від одного на одному і тому ж матеріалі навчаються кілька машин. При впізнанні нових об'єктів кожна машина буде відносити ці об'єкти до якогось

образу, може статися так, що не до одного і того ж. Остаточне рішення приймається "голосуванням" машин - об'єкт ставитися до того образу, до якого його віднесло більше число машин.

Спосіб підвищення надійності розпізнавання полягає в деякому поліпшенні методу проведення січних площин. Можна припустити, що якщо проводити січні площини близько до площини, що проходить через середину прямої, що з'єднує об'єкт і об'єкт, перпендикулярний цій прямій, то результуюча поверхня буде ближче до істинного кордону між образами. Експерименти підтверджують це припущення.

В алгоритмі, заснованому на методі потенціалів, з кожним потурбованим елементом поля рецепторів можна зв'язати деяку функцію, рівну одиниці на цьому елементі і спадну в усіх напрямках від нього, тобто функцію ϕ , аналогічну електричного потенціалу з тією лише різницею, що в даному випадку R є відстань між двома сусідніми елементами поля рецепторів [37].

$$\phi(R) = \frac{1}{1 + \alpha R^2}$$

Для підрахунку користуються таким правилом: кожен потурбований елемент поля рецепторів має "власний" потенціал, рівний одиниці, і який в свою чергу збільшує на $\frac{1}{2}$ потенціали всіх (в тому числі і потурбованих) сусідніх з ним елементів по горизонталі, вертикалі і діагоналях. Однак цей метод кодування може бути поліпшений. Якщо пов'язати з кожним потурбованим елементом поля рецепторів деяку функцію, рівну одиниці на цьому елементі і спадну в усіх напрямках від нього, тобто функцію, аналогічну потенціалу ϕ , з тією лише різницею, що в даному випадку R є відстань між двома сусідніми елементами поля рецепторів (рис. 2.6).

2	2,5	2	2	2	1,5	
2,5	2,5	1,5	1,5	1,5	1	
2	1,5	0	0	0	0	
2,5	2,5	1,5	1,5	1	0,5	
2	2,5	2	2	2	1	
1	1,5	1,5	1,5	2	2	
0	0	0	0	1,5	2	
0,5	0,5	0	0	1,5	2	
1,5	1,5	1,5	1,5	2	2	
1	2	2	2	2	1	

2	2,5	2,5	2	2	1,5	
2	3	2,5	1,5	1,5	1	
1,5	2	1,5	0	0	0	
2	3	2,5	1,5	1	0,5	
2	2,5	2,5	2	2	1	
1	1,5	1,5	1,5	2	3	
0	0	0	0	1,5	2	
0,5	0,5	0	0	1,5	2	
1,5	1,5	1,5	1,5	2	2	
1	2	2	2	2	1	

1,5	2	2	2	2,5	2	
1	1,5	1,5	1,5	2,5	2,5	
0	0	0	0	1,5	2	
1	1,5	1,5	1,5	2	2	
1,5	2	2	2	2	1,5	
1	1,5	1,5	1,5	2	2	
0	0	0	0	1,5	2	
0,5	0,5	0	0	1,5	2	
1,5	1,5	1,5	1,5	2	2	
1	2	2	2	2	1	

Рис. 2.6 – Метод потенціалів

Найпростіший алгоритм впізнавання, побудований на методі потенціалів, можна здійснити в два етапи:

- 1 Навчання (в процесі навчання запам'ятовуються коди всіх точок і вказівок, до якого з образів відноситься кожна точки).
- 2 Впізнавання (в процесі пізнавання проводиться ідентифікація і видається інформація, до якого образу належить закодована матриця).

Незважаючи на деякі обмеження вихідної форми моделі персептрона Розенблатта, вона стала основою для багатьох сучасних найбільш складних алгоритмів навчання з учителем [30]. Прикладом персептрона з нерекурентною мережею може служити модель, показана на рис. 2.8. Вона використовує алгоритм навчання з учителем; іншими словами, навчальна вибірка складається з множини вхідних векторів, для кожного з яких зазначено свій необхідний вектор цілі. Компоненти вхідного вектора представлені безперервним діапазоном значень; компоненти вектора мети є двійковими величинами (0 або 1). Після навчання мережа отримує на вході набір безперервних входів і виробляє необхідний вихід у вигляді вектора з бінарними компонентами [35].

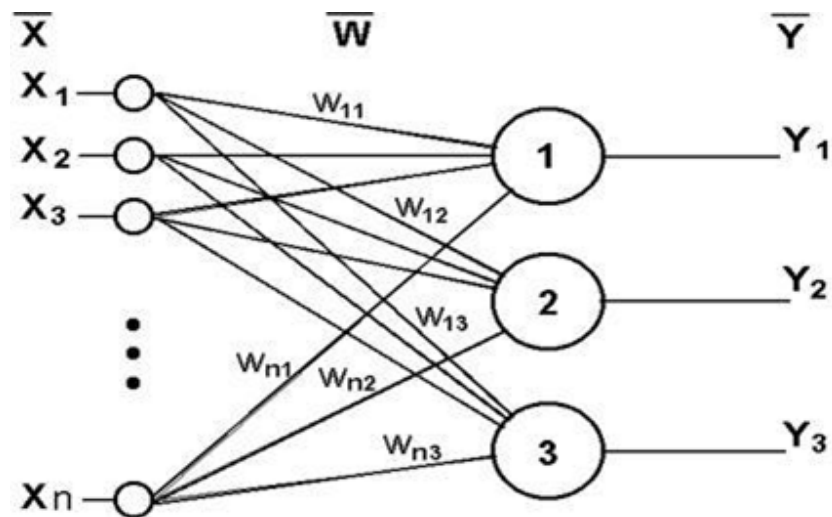


Рис. 2.7 - Багатошарова нейронна мережа

Навчання здійснюється наступним чином:

1. Рандомізуються всі ваги мережі в малі величини.
2. На вхід мережі подається вхідний навчальний вектор X і обчислюється сигнал NET від кожного нейрона, використовуючи стандартний вираз.

$$NET_j = \sum_i xw$$

3. Обчислюється значення порогової функції активації для сигналу NET від кожного нейрона.
4. Обчислюється помилка для кожного нейрона за допомогою віднімання отриманого виходу з необхідного виходу:

$$error_j = target_j - OUT_j$$

5. Кожна вага модифікується в такий спосіб:

$$W_{ij}(t+1) = w_{ij}(t) + \alpha x_j error_j$$

6. Повторюються кроки з другого по п'ятий доти, поки помилка не стане досить малою.

Персептрон Розенблатта обмежується бінарними виходами. Удвоу разом з Хоффом розширили алгоритм навчання персептрона на випадок

безперервних виходів, використовуючи сигмоїдальну функцію. Вони розробили математичний доказ того, що мережа при певних умовах буде сходиться до будь-якої функції, яку вона може уявити. Їх перша модель - Адалін має один вихідний нейрон, більш пізня модель – Мадаліна, розширює її на випадок з багатьма вихідними нейронами.

Вирази, що описують процес навчання Адалін, дуже схожі з персептронами. Істотні відмінності є в четвертому кроці, де використовуються безперервні сигнали *NET* замість бінарних *OUT*. Модифікований крок 4 в цьому випадку реалізується в такий спосіб [24]: Обчислюється помилка для кожного нейрона за допомогою віднімання отриманого виходу з необхідного виходу.

Результати досліджень Кохонена на самоорганізованих структурах, для задач розпізнавання і класифікації образів, представлені векторними величинами, в яких кожен компонент вектора відповідає елементу образу. Алгоритми Кохонена ґрунтуються на техніці навчання без учителя. Після навчання подача вхідного вектора з даного класу буде приводитися до вироблення турбуючого рівня в кожному вихідному нейроні; нейрон з максимальним порушенням представляє класифікацію. Так як навчання проводиться без вказівки цільового вектора, то немає можливості визначати заздалегідь, який нейрон буде відповідати даному класу вхідних векторів. Проте, це планування легко проводиться шляхом тестування мережі після навчання.

Алгоритм трактує набір з n вхідних ваг нейрона як вектор в n -вимірному просторі. Перед навчанням кожен компонент цього вектору ваг ініціалізується в випадковою величиною. Потім кожен вектор нормалізується в вектор довжиною в один символ в просторі ваг. Це робиться діленням кожного випадкової ваги на квадратний корінь з суми квадратів компонент цього вагового вектору.

Всі вхідні вектору навчального набору також нормалізуються, і мережа навчається згідно з наступним алгоритмом [24]:

1. Вектор X подається на вхід мережі.
2. Визначаються відстані D_j (в n -вимірному просторі) між X і ваговими векторами W_i кожного нейрона. В евклідовому просторі це відстань обчислюється за такою формулою

$$D_j = \sqrt{\sum_i (x_i - w_i)^2}$$

3. Нейрон, який має ваговий вектор, найближчий до X , оголошується переможцем. Ваговий вектор, названий W_c , стає основним в групі вагових векторів, які лежать в межах відстані D від W_c .
4. Група вагових векторів налаштовується у відповідності з наступним виразом:

$$W_j(t+1) = W_j(t) + \alpha[X - W_j]$$

5. Повторюються кроки з 1 по 4 для кожного вхідного вектору.

У процесі навчання нейронної мережі значення D і α поступово зменшуються. Коефіцієнт α на початку навчання промені встановлювати приблизно рівним 1 і зменшувався в процесі навчання до 0, в той час як D може на початку навчання дорівнювати максимальній відстані між ваговими векторами і в кінці навчання стати настільки малим, що буде навчатися тільки один нейрон.

Точність класифікації буде поліпшуватися при додатковому навчанні. Згідно з рекомендацією Кохонена, для отримання хорошої статистичної точності кількість навчальних циклів повинна бути, принаймні, в 500 разів більше кількості вихідних нейронів.

Навчальний алгоритм налаштовує вагові вектори в околиці порушеного нейрона таким чином, щоб вони були більш схожими на вхідний вектор. Так як всі вектори нормалізуються в вектори довжиною в один символ, вони можуть розглядатися як точки на поверхні одиничної гіперсфери. У процесі навчання група сусідніх вагових точок переміщується ближче до точки вхідного вектора.

Передбачається, що вхідні вектори фактично групуються в класи відповідно до їх становища у векторному просторі [21]. Певний клас буде асоціюватися з певним нейроном, переміщаючи його ваговий вектор в напрямку центру класу і сприяючи його порушення при появі на вході будь-якого вектора даного класу. Після навчання класифікація виконується за допомогою подачі на вхід мережі випробуваного вектора, обчислення турбування для кожного нейрона з подальшим вибором нейрона з найвищим рівнем турбування як індикатора правильної класифікації.

2.3 Модель розпізнавання образів

Великий поштовх розвитку нейрокібернетики дав нейрофізіолог Френк Розенблат, який запропонував модель розпізнавання образів, яку назвав "Персептрон" (від латинського *percepto* - розумію, пізнаю) [36]. При її розробці він опирався з деяких прийнятих уявлень про структуру мозку і зорового апарату. Прагнучи відтворити функції людського мозку, він використовував просту модель біологічного нейрона (рис. 2.8) і систему зв'язків між ними.

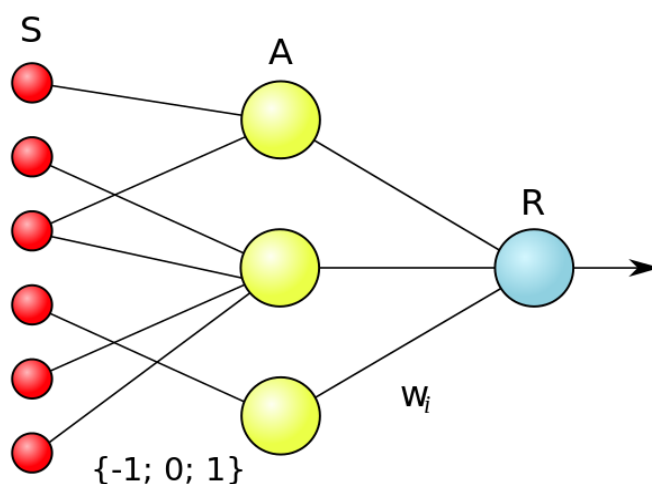


Рис. 2.8 – Персептронний нейрон

Пристроєм "чуттів" персептрона служить фотоелектрична модель сітківки - поле рецепторів, що складається з декількох сотень фоторезисторів (S-елементів) (рис. 2.9). Кожен елемент поля рецепторів може перебувати в двох станах - збудженому або стані спокою, в залежності від того, падає чи ні

на відповідне фоторезистор контур проектованої на поле фігури. На виході кожного елемента з'являється сигнал x_i ($i = 1, 2, \dots, n$, де n - число елементів), що дорівнює одиниці, якщо елемент збуджений, і нулю - в іншому випадку. Наступною частиною персептрона служать, так звані, асоціативні елементи або А-елементи. Кожен А-елемент має декілька входів і один вихід. При підготовці персептрона до експерименту до входів А-елемента підключаються виходи рецепторів, причому підключення будь-якого з них можна виготовити зі знаком плюс або зі знаком мінус.

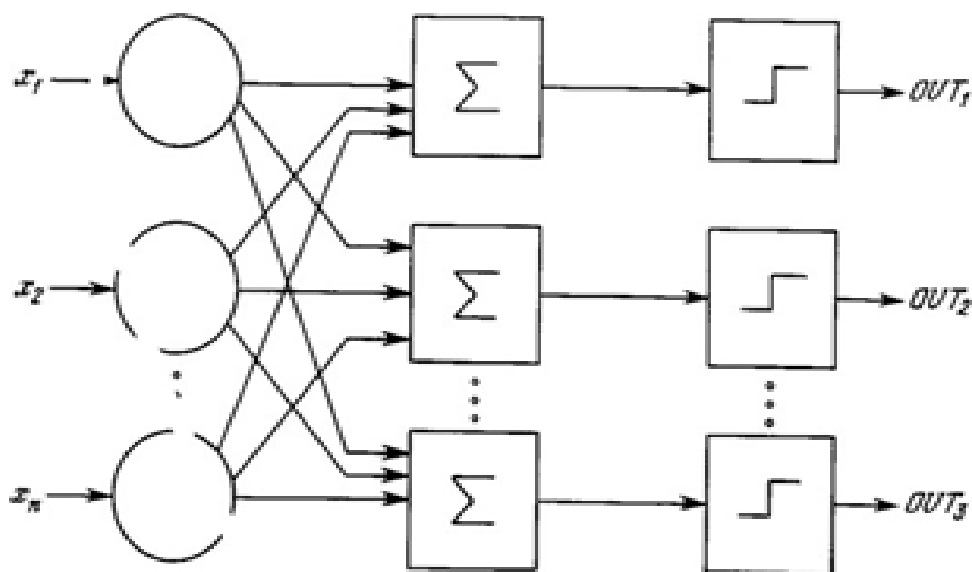


Рис. 2.9 – Персептронний нейрон з багатьма виходами

Вибір рецепторів, що підключаються до даного А-елементу, також як і вибір знаку підключення, проводиться випадково. В ході експерименту зв'язок рецепторів з А-елементами залишається незмінним. А-елементи виробляють алгебраїчне підсумовування сигналів [36], що надійшли на їх входи, і отриману суму порівнюють з однаковою для всіх А-елементів величиною θ .

Якщо сума більше θ , А-елемент турбується і видає на виході сигнал, що дорівнює одиниці. Якщо сума менше θ , А-елемент залишається стурбованим і вихідний його сигнал дорівнює нулю [36]. Таким чином, вихідний сигнал j -го А-елемента, де величина r_{ij} приймає значення $+1$, якщо i -й рецептор підключений до входу j -го А-елемента зі знаком плюс; і значення -1 , якщо

рецептор підключений зі знаком мінус, і значення 0, якщо i -й рецептор до j -го А-елементу не може підключитися ($j = 1, 2, \dots, m$, де m - число А-елементів). Вихідні сигнали А-елементів за допомогою спеціальних пристроїв (підсилювачів) множаться на змінні коефіцієнти λ_j . Кожен з цих коефіцієнтів може бути позитивним, негативним або рівним нулю і змінюватися незалежно від інших коефіцієнтів. Вихідні сигнали підсилювачів підсумовуються, і сумарний сигнал надходить на вхід, так званого, елемента реакції або R-елемента. Якщо величина σ позитивна або дорівнює нулю, R-елемент видає на виході одиницю, а якщо σ негативна – нуль.

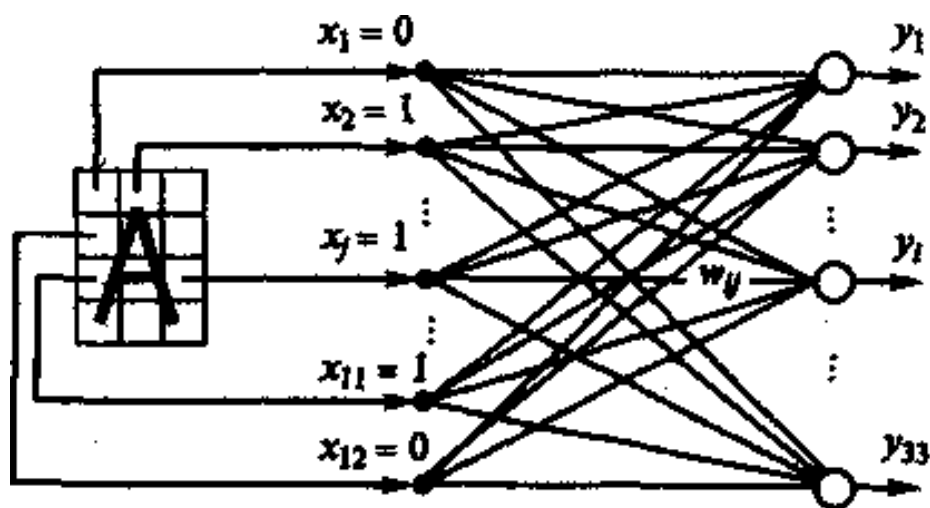


Рис. 2.10 – Персептрон з декількома виходами

Припустимо, що на поле рецепторів проектується фігури, що належать до двох різних образів. Якщо виявиться можливим привести персептрон в такий стан, щоб він з достатньою надійністю видавав на виході 1, при появі на його вході фігур одного образу, то це буде означати, що персептрон має здатність навчатися розрізняти два образи [36].

Описана структура персептрона дозволяє розділяти запропоновані об'єкти тільки на дві множини. Для розпізнавання більшого числа образів, наприклад, трьох А, В і С може бути застосований персептрон, побудований за схемою (рис. 2.10). Вихідний сигнал кожного А-елемента надходить не на один, а на кілька (за кількістю образів) підсилювачів. Після множення на λ

вихідні сигнали надходять на суматори Σ , кількість яких також дорівнює числу різних образів. Замість R-елемента встановлено пристрій, що порівнює між собою вихідні сигнали суматорів. Пред'явлений об'єкт відноситься до того образу, акумулятор якого має найбільший сигнал.

Для розпізнавання декількох образів може бути використаний перцептрон дещо іншої структури. У такому перцептроні A-елементи розбиті на кілька груп, кожна з яких пов'язана зі своїм суматором і R-елементом. Сукупність вихідних сигналів R-елементів можна розглядати як виражений в двійковому коді номер образу, що і дає такому перцептрону можливість розбивати об'єкти на кілька класів. Наприклад, для класифікації на вісім класів досить трьох груп. У цьому випадку можливі наступні вісім комбінацій вихідних сигналів трьох R- елементів: 000, 001, 010, 011, 100, 101, 110, 111. Поява кожної з цих комбінацій можна розглядати як віднесення перцептроном пред'явленої йому фігури до одного з восьми образів.

Кожна з груп A-елементів, з'єднаних зі своїм R-елементом, за структурою ідією цілком аналогічна перцептрону, здатному розбивати об'єкти на два класи. Навчання перцептрона складається з ряду послідовних тактів. У кожному такті перцептрону пред'являється об'єкт одного з образів. Залежно від реакції перцептрона на пред'явлену йому фігуру за певними правилами проводиться зміна коефіцієнтів λ_j . Виявляється можливим за деяку кінцеву кількість тактів привести перцептрон в такий стан, що він з достатньою впевненістю розпізнає фігури, що йому пред'являються.

Можливі два типи алгоритмів навчання перцептрона. Перший з них не враховує правильності відповідей перцептрона в процесі навчання, і зміна λ_j в кожному такті проводиться незалежно від того, "дізнався" або не "дізнався" перцептрон пред'явлену в цьому такті фігуру. В алгоритмах другого типу коефіцієнти λ_j змінюються з урахуванням правильності відповідей перцептрона.

Алгоритм першого типу здійснюється наступним чином. Заздалегідь встановлюється, що після навчання перцептрон повинен видавати на виході: 1

– при відповідності, наприклад, об'єктів образу A , і 0 - при пред'явленні образу B . Потім пред'являють персептрон об'єкти кожного з образів. У кожному такті персептрон відповідає на пред'явлений йому об'єкт турбуванням деяких A -елементів. Навчання полягає в тому, що коефіцієнти λ_j порушених у даному такті A -елементів збільшуються на деяку величину (наприклад на одиницю), якщо в цьому такті був пред'явлений об'єкт образу A , і зменшується на цю ж величину, якщо був пред'явлений об'єкт образу B . Природно, що така зміна коефіцієнтів λ_j повинна призводити до підвищення правильності відповідей персептрона, так як збільшення λ_j порушених A -елементів призводить до збільшення сигналу на вході R -елемента, а їх зменшення - до зменшення сигналу. Відповідно до прийнятих умов персептрон буде давати правильні відповіді, якщо образу A будуть відповідати позитивні, а образу B - негативні сигнали на вході R -елементу.

При розробці персептрона Ф. Розенблат намагався моделювати деякі властивості живого мозку [36]. Персептрон або будь-яка програма, що імітує процес розпізнавання, працюють в двох режимах: в режимі навчання і в режимі розпізнавання.

Відомо, що мозок здатний зберігати або відновлювати багато своїх функцій при серйозних пошкодженнях, викликаних травмами або захворюваннями. Стійкість персептрона до порушень його структури має певну схожість з цією властивістю мозку. З усіх цих міркувань не можна зробити висновок, що алгоритми мозку і персептрона збігаються. Однак в даний час персептрон є, мабуть, найбільш правдоподібною моделлю мозку.

Здатність штучних нейронних мереж навчатися є їх найбільш важливою властивістю. Подібно біологічним системам, які вони моделюють, ці нейронні мережі моделюють самі себе в результаті спроб досягти кращої моделі поведінки [36].

Алгоритм навчання персептрона можна реалізувати на цифровому комп'ютері або іншому електронному пристрої, завдяки чому мережа набуває

властивості самоналаштування. Тому процес коригування ваг зазвичай називається "навчанням", і говорять, що мережа "вчиться".

Доказ Розенблатта став основною віхою і надав потужний імпульс дослідженням у цій галузі.

2.4 Висновки до розділу 2

У розділі досліджено проблеми, які виникають при комп'ютерному моделюванні розпізнавання образів. Розглянуто підходи до вирішення цих проблем, відштовхуючись від аналогії з біологічними процесами. Наведено приклади і порівняння алгоритмічних побудов класифікаторів штучної нейронної мережі. Детально досліджено персептрон, що є одним з основних структурних комп'ютерних моделей сприйняття інформації мозком, як модель розпізнавання.

РОЗДІЛ 3

РОЗРОБКА ВЕБДОДАТКА ДЕТЕКТОРА МАСОК

У цьому розділі розглянуто розробку та тренування (навчання) детектора маски для обличчя за допомогою Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask.

Надамо коротку характеристику програмним інструментам та технологіям розробки проєкту:

- *Python* - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією [14]. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована. Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

- *OpenCV* (Open Source Computer Vision Library) - бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях, відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях. Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору [11].

– *Keras/TensorFlow*. Keras - одна з найпотужніших і простих у використанні бібліотек Python для розробки та оцінки моделей глибокого навчання. Він включає в себе бібліотеки ефективних чисельних обчислень Theano і TensorFlow. Основна перевага Keras в тому, що можна легко розпочати роботу з нейронними мережами [10]. Для реалізації проєкту потрібна TensorFlow - це комплексна платформа з відкритим вихідним кодом для машинного навчання, що має всеосяжну гнучку екосистему інструментів, бібліотек і ресурсів спільноти, яка дозволяє легко створювати і розгортати додатки на основі машинного навчання [15].

– *Deep Learning* (глибинне навчання) - це галузь машинного навчання, що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома шарами обробки, що побудовано з кількох лінійних або нелінійних перетворень [8].

– *Flask* - мікрофреймворк для веб-додатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2, не вимагає спеціальних засобів чи бібліотек має підтримку розширень, які надають додаткові властивості [5].

У роботі використовується двофазний детектор маски для обличчя. Детектор маски потенційно може бути використаний для забезпечення персональної та соціальної безпеки для прийняття рішення про впровадження заходів з перешкоджання розповсюдженню хвороби. Надалі буде докладно описано впровадження конвеєру для комп'ютерного зору/глибинного навчання, що є основною складовою детектора.

Для того, щоб навчити і розгорнути в вигляді вебдодатку детектор маски для обличчя, потрібно розбити проєкт на дві різні фази, кожна з яких має свої відповідні етапи (як показано на рис. 3.1).

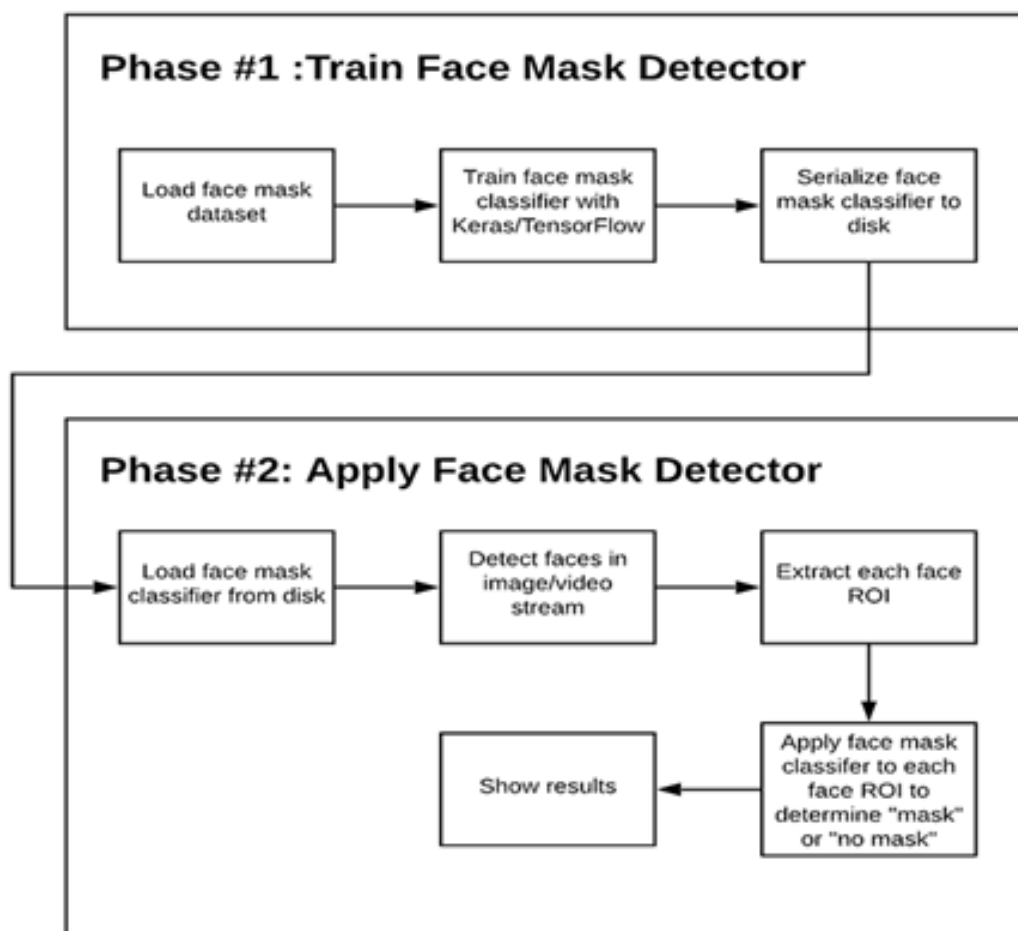


Рис. 3.1 – Фази і етапи для побудови детектора маски для обличчя

1. *Навчання:* Завантаженні набору виявлення маски обличчя з носія, навчанні моделі (за допомогою Keras / TensorFlow) на цьому наборі даних, а потім серіалізації детектора маски обличчя на носія.
2. *Розгортання:* Після того, як детектор маски обличчя пройшов навчання, ми можемо перейти до завантаження детектора маски, виконуючи виявлення обличчя, а потім класифікуючи кожне обличчя як
 - with_mask - з маскою;
 - without_mask - без маски.

Детально розглянемо кожну з цих фаз та пов'язані з ними підмножини.

3.1 Підготовка набору даних та навчання ML моделі

Набір даних, який використовується у проєкті для навчання детектора маски обличчя було взяти з вільного доступу Google Colab [23]. Набір даних

виявлення маски на обличчі складається із зображень “з маскою” та “без маски”.

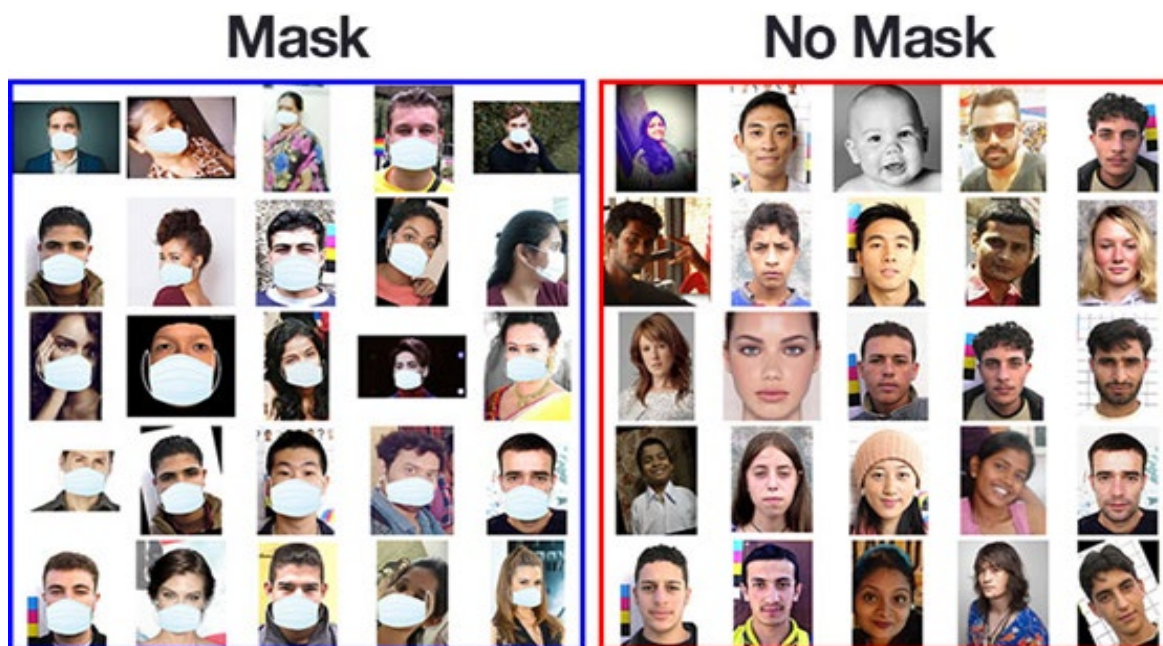


Рис.

3.2 – Приклади з набору даних

Набір складається з 1376 зображень, що належать до двох класів:

- with_mask: 690 зображень;
- without_mask: 686 зображень.

Мета полягає в тому, щоб підготувати глибинну модель користувацького навчання для визначення наявності маски на людині.

Для створення набору даних було виконано наступне:

1. Знято/знайдено звичайні зображення обличь;
2. Створено скрипт Python з використанням комп'ютерного зору, необхідний для додавання до зображень маски на обличчя, створюючи тим самим штучний (але все ще застосовний в реальному світі) набір даних.

Орієнтири на обличчі дозволяють нам автоматично робити висновки про розташування лицевих структур, включаючи:

- Очі;

- Брови;
- Ніс;
- Рот;
- Щелепа.

Щоб використовувати орієнтири обличчя для побудови набору даних про обличчя, які носять маски для обличчя, спочатку слід почати із зображення людини, яка не носить маску (рис. 3.3).



Рис. 3.3 – Приклад людини, що не носить маску.

Для обчислення розташування обмежувальної рамки обличчя на зображенні застосовується розпізнавання обличчя з використанням методу глибинного навчання за допомогою OpenCV (рис. 3.4).



Рис. 3.4 – Приклад розпізнавання обличчя

Як тільки стає відомо, де на зображенні знаходиться обличчя, ми можемо витягти область інтересу обличчя (Region of Interest - ROI), тобто ту частину зображення яка нам необхідна для подальшої роботи (рис. 3.5).



Рис. 3.5 – Вилучення ROI на обличчі за допомогою OpenCV та NumPy

Після отримання необхідної області знаходяться і застосовуються орієнтири обличчя (face landmarks), дозволяючи локалізувати очі, ніс, рот тощо (рис. 3.6).



Рис. 3.6 – Знаходження і застосування орієнтирів обличчя

Для подальшої розробки генератора необхідно мати зображення маски (з прозорим фоном) (рис.3.7). Ця маска для обличчя буде автоматично накладена на початкову рентабельність інвестицій, оскільки ми знаємо місця розташування орієнтирів на обличчі.



Рис. 3.7 – Приклад маски/щитка для захисту

Ця маска буде автоматично нанесена на обличчя, використовуючи орієнтири на обличчі (а саме точки вздовж підборіддя та носа), щоб визначити, де буде розміщена маска. Потім маска змінюється в розмірі і обертається для накладення її на обличчя. На рис. 3.8 маска зображено приклад розміщення на обличчі людини маски. З першого погляду важко сказати, що маска була штучно накладена з OpenCV та dlib face.



Рис. 3.8 – Приклад штучно створеного об'єкту за набору даних

Потім можна повторити цей процес для всіх вхідних зображень, тим самим створивши набір штучних масок для обличчя: Отриманий набір стане частиною набору даних, що буде використовуватися у проекті, "з

маскою"/"без маски" для виявлення маски для обличчя за допомогою комп'ютерного зору та глибинного навчання.



Рис. 3.9 – Штучний набір зображень маски для обличчя

Є нюанс, який варто зазначити, при використанні подібного методу для штучного створення набору даних - не можна повторно використовувати зображення, яке використовувалося у штучному процесі генерації.

Якщо використовувати оригінальні зображення, що були застосовані для генерації зразків, модель стане сильно упередженою і не зможе добре узагальнити. Потрібно уникати цього і збирати нові приклади.

3.2 Розгортання застосунку та приклади роботи

Перед створенням веб-версії додатку, який в режимі реального часу виконує розпізнавання обличчя з перевіркою наявності маски на ньому, модель машинного навчання було протестовано на декількох прикладах статичних зображень (рис. 3.10).



Рис. 3.10 – Робота додатку на прикладі зображень

При задовільних вхідних даних на зображенні рамка визначення обличчя фарбується зеленим з відповідною поміткою "Mask". Навпаки, при незбіжності рамка фарбується красним з поміткою "Not Mask".

Для розгортання і створення додатку, що в режимі реального часу обробляє дані з веб-камери було використано фреймворк Flask. Flask ідеально підходить під виконання поставленої задачі, так як дозволяє швидко і без довготривалих налаштувань створити і розгорнути малі та середні за об'ємом веб-програми.

3.3 Висновки до розділу 3

Враховуючи натренований детектор маски для обличчя, було впроваджено додаткові скрипти Python, які використовуються для:

1. Виявлення маски для обличчя на зображеннях;
2. Виявлення маски для обличчя у потоках відео в режимі реального часу.

У розділі досліджено процес формування даних необхідних для тренування детектора маски, представлено процес розпізнавання і розглянуто розроблений додаток.

Проведено моделювання двофазної архітектури, яка дозволяє структуровано навчити і розгорнути в вигляді веб додатку детектор маски для обличчя.

ВИСНОВКИ

У рамках випускної кваліфікаційної роботи було виконано загальний аналіз сучасного стану розглянутої проблеми. На основі проведених досліджень та аналізу і обробки спеціальної літератури та ресурсів Інтернету були опрацьовано, виявлено і застосовано принципи і знання з теорії нейробіології, кібернетики, штучного інтелекту, штучних нейронних мереж і нейромереж самоорганізації, перцептронів, машинного розпізнавання образів.

Приділено увагу особливостям класифікації графічних образів, наведено алгоритмічні побудови та рецепторну структуру сприйняття інформації.

Надано огляд штучної нейронної мережі, як інструменту для вирішення задачі. Досліджено проблеми, які виникають при комп'ютерному моделюванні розпізнавання образів. Розглянуто підходи до вирішення цих проблем, відштовхуючись від аналогії з біологічними процесами. Наведено приклади і порівняння алгоритмічних побудов класифікаторів штучної нейронної мережі. Детально досліджено перцептрон, що є одним з основних структурних комп'ютерних моделей сприйняття інформації мозком, як модель розпізнавання.

Реалізовано засобами Python, OpenCV, Keras/TensorFlow, Deep Learning та Flask вебдодаток детектора маски на обличчі. Впроваджено додаткові скрипти Python для можливостей виявлення маски на обличчі на зображеннях та у потоках відео в режимі реального часу. Проведено моделювання двофазної архітектури, яка дозволяє структуровано навчити і розгорнути в вигляді веб додатку детектор маски для обличчя. Досліджено процес формування даних необхідних для тренування детектора маски, представлено процес розпізнавання розробленим вебдодатком.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AEI: Artificial ‘Emotional’ Intelligence. URL: <https://towardsdatascience.com/aei-artificial-emotional-intelligence-ea3667d8ece>. (Date of access: 4.12.2024).
2. Ekman P., Darwin and Facial expressions, Academic Press, 1973. pp. 169-222.
3. Ekman P., Facial expression and emotion, American Psychologist, 1993. pp.48:384-392,
4. Emotion Detection From Facial Expressions.. URL: <https://www.kaggle.com/c/emotion-detection-from-facial-expressions/overview> (Date of access: 4.11.2024).
5. Flask’s. URL: <https://flask.palletsprojects.com/en/2.0.x/>. (дата звернення: 10.11.2024).
6. Handling imbalanced datasets in machine learning. URL: <https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28>. (Date of access: 4.11.2024).
7. How to Develop a Cost-Sensitive Neural Network for Imbalanced Classification. 2020. URL: <https://machinelearningmastery.com/cost-sensitive-neural-network-for-imbalanced-classification/> (Date of access: 4.11.2024).
8. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press. (2016). URL: <https://www.deeplearningbook.org/>. (дата звернення: 14.11.2024).
9. Jason Brownlee. How to Use The Pre-Trained VGG Model to Classify Objects in Photographs. URL: <https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/> (Date of access: 4.11.2024).
10. Karlijn Willems, Keras Tutorial: Deep Learning in Python. December 10th, 2019. URL: https://www.datacamp.com/community/tutorials/deep-learning-python?utm_source=adwords_ppc&utm_campaignid=1658343524_wcB. (дата

звернення: 10.11.2024).

11. OpenCV 3.2. URL: <https://opencv.org/opencv-3-2/>. (дата звернення: 10.11.2024).

12. OpenCV-Python Tutorials URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html (Date of access: 4.11.2024).

13. Pedro Domingos, The Master Algorithm, Publisher: INGRAM PUBLISHER SERVICES US. 2018. 352p.

14. Python core development news and information. URL: <https://blog.python.org/2021/01/python-3100a4-is-now-available-for.html>. (дата звернення: 10.11.2024).

15. TensorFlow. URL: <https://www.tensorflow.org/>. (дата звернення: 10.11.2024).

16. Understanding the VGG19 Architecture. URL: <https://iq.opengenus.org/vgg19-architecture/>. (Date of access: 4.11.2024).

17. VGG16 architecture. URL: <https://iq.opengenus.org/vgg16/> (Date of access: 4.11.2024).

18. The State of Machine Learning Frameworks in 2019. URL: <https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/> (дата звернення: 29.11.2024).

19. Гончаренко М.О. Сравнительный анализ методов формирования дескрипторов изображений в контексте задачи сегментации видеопотока. *Бионика интеллекта*. 2015. № 2 (85). С. 90–94

20. Козуб Г.О., Козуб Ю.Г. Методичні рекомендації до виконання кваліфікаційних робіт за напрямом 122 Комп'ютерні науки за освітнім рівнем „магістр”. // Старобільськ: ДЗ «ЛНУ імені Тараса Шевченка», 2021.— 70с.

21. Жерон Орельен «Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow» Вильямс, 2018 год, 688 стр

22. Haar A. Zur Theorie der orthogonalen Funktionensysteme, *Mathematische Annalen*, 1910, 69, pp. 331–371.

23. Інструкція по роботі з Google Colab. URL: <https://github.com/deepmipt/dlschl/wiki/Инструкция-по-работе-с-Google-Colab> (Date of access: 4.11.2024).
24. К.В. Воронцов, Математические методы обучения по прецедентам (теория обучения машин), 2015. 141с.
25. Каскад Хаара. URL: <http://wiki.amplab/cgi-bin/awki.cgi/КаскадХаара>. (дата звернення: 30.11.2024).
26. Разинкин В. Б., Катерміна Т. С. Розпізнавання обличчя за фотографією. Міжнародний журнал передових досліджень. 2018. №1-2. С. 171-180.
27. Джулли А., Пал С. Библиотека Keras — инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow = Deep learning with Keras. ДМК-Пресс, 2017. 294 с.
28. Машинное обучение. URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение (дата звернення: 30.11.2024).
29. Местецкий Л.М. Математические методы распознавания образов, М.: МГУ, ВМиК, 2002–2004. – 85 с.
30. Наконечний А.Й. Цифрова обробка сигналів/ Навчальний посібник / А.Й. Наконечний, Р.А. Наконечний, В.А. Павлиш. Львів: Видавництво Львівської політехніки, 2010. 368 с.
31. Оліх В.Я. Використання нейронних мереж для вирішення задач розпізнавання образів / В.Я. Оліх, А.І. Сегін // Матеріали проблемно-наукової міжгалузевої конференції. – Надвірна – Яремче, Україна – 2016. – С.117-120.
32. Понимание сверточных нейронных сетей через визуализации в PyTorch. URL: <https://habr.com/ru/post/436838/>. (дата звернення: 30.11.2024).
33. Построение SIFT дескрипторов и задача сопоставления изображений URL: <https://habr.com/ru/post/106302/> (дата звернення: 29.11.2024).
34. Р. Гонсалес, Р. Вудс. Цифровая обработка изображений, М.: Техносфера, 2005.– 1072 с.

35. Распознавание изображений. Алгоритм Eigenface. URL: <https://habr.com/ru/post/68870/>. (дата звернення: 30.11.2024).
36. Розенблатт Ф. Принципы нейродинамики: перцептроны и теория механизмов мозга / Пер. с англ. В.Я. Алтаева, Б.А. Власюкова, Ю.А. Крутикова, Ю.А. Патругина. М.: Мир, 1965. 478 с.
37. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество. URL: <https://habr.com/ru/post/348000/>. (дата звернення: 30.11.2024).
38. Урядовий портал. Оперативна інформація щодо заходів, які вживаються іноземними країнами з протидії розповсюдженню COVID-19. URL: <https://mfa.gov.ua/news/operativna-informaciya-shchodo-zahodiv-yaki-vzhivayutsya-inozemnimi-krayinami-z-protidiyi-rozpovsyudzhennyu-covid-19>(дата звернення: 29.11.2024).
39. Як працює згорткова нейронна мережа (CNN). Базовий курс. (2018). URL: <https://neurohive.io/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/>(дата звернення: 30.11.2024).

ДОДАТКИ

Додаток А. Код app.py

```

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
from flask import Response
from flask import Flask
from flask import render_template
import threading
import datetime
import imutils

outputFrame = None
lock = threading.Lock()
app = Flask(__name__, template_folder="templates")
vs = VideoStream(src=0).start()
time.sleep(2.0)

@app.route("/")
def index():
    # return the rendered template
    return render_template("index.html")

print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join(["face_detector", "deploy.prototxt"])
weightsPath = os.path.sep.join(
    ["face_detector", "res10_300x300_ssd_iter_140000.caffemodel"]
)
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

print("[INFO] loading face mask detector model...")
maskNet = load_model("mask_detector.model")

def detect_and_predict_mask(frame):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300), (104.0, 177.0, 12
3.0))

    faceNet.setInput(blob)
    detections = faceNet.forward()

```

```

faces = []
locs = []
preds = []

for i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]

    if confidence > 0.5:
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

        face = frame[startY:endY, startX:endX]
        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (224, 224))
        face = img_to_array(face)
        face = preprocess_input(face)

        faces.append(face)
        locs.append((startX, startY, endX, endY))

if len(faces) > 0:
    faces = np.array(faces, dtype="float32")
    preds = maskNet.predict(faces, batch_size=32)

return (locs, preds)

def detect():
    global vs, outputFrame, lock
    while True:
        frame = vs.read()
        frame = imutils.resize(frame, width=400)

        (locs, preds) = detect_and_predict_mask(frame)

        for (box, pred) in zip(locs, preds):
            (startX, startY, endX, endY) = box
            (mask, withoutMask) = pred

            label = "Mask" if mask > withoutMask else "No Mask"
            color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

            label = "{:}: {:.2f}%".format(label, max(mask, withoutMask) * 1
00)

```

```

        cv2.putText(
            frame,
            label,
            (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.45,
            color,
            2,
        )
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    with lock:
        outputFrame = frame.copy()

def generate():
    global outputFrame, lock

    while True:
        with lock:
            if outputFrame is None:
                continue

            (flag, encodedImage) = cv2.imencode(".jpg", outputFrame)

            if not flag:
                continue

            yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +
                bytearray(encodedImage) + b'\r\n')

@app.route("/video_feed")
def video_feed():
    return Response(generate(),
                    mimetype="multipart/x-mixed-replace; boundary=frame")

if __name__ == '__main__':
    t = threading.Thread(target=detect)
    t.daemon = True
    t.start()

    app.run(host="0.0.0.0", port=8000, debug=True,
            threaded=True, use_reloader=False)

```

Додаток Б. Код mask.py

```

import os
import sys
import random
import argparse
import numpy as np
from PIL import Image, ImageFile

IMAGE_DIR = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'images')
DEFAULT_IMAGE_PATH = os.path.join(IMAGE_DIR, 'default-mask.png')
BLACK_IMAGE_PATH = os.path.join(IMAGE_DIR, 'black-mask.png')
BLUE_IMAGE_PATH = os.path.join(IMAGE_DIR, 'blue-mask.png')
RED_IMAGE_PATH = os.path.join(IMAGE_DIR, 'red-mask.png')

def cli():
    parser = argparse.ArgumentParser(description='Wear a face mask in the given picture.')
    parser.add_argument('pic_path', help='Picture path.')
    parser.add_argument('--show', action='store_true', help='Whether show picture with mask or not.')
    parser.add_argument('--model', default='hog', choices=['hog', 'cnn'], help='Which face detection model to use.')
    group = parser.add_mutually_exclusive_group()
    group.add_argument('--black', action='store_true', help='Wear black mask')
    group.add_argument('--blue', action='store_true', help='Wear blue mask')
    group.add_argument('--red', action='store_true', help='Wear red mask')
    args = parser.parse_args()

    pic_path = args.pic_path
    if not os.path.exists(args.pic_path):
        print(f'Picture {pic_path} not exists.')
        sys.exit(1)

    if args.black:
        mask_path = BLACK_IMAGE_PATH
    elif args.blue:
        mask_path = BLUE_IMAGE_PATH
    elif args.red:
        mask_path = RED_IMAGE_PATH
    else:
        mask_path = DEFAULT_IMAGE_PATH

    FaceMasker(pic_path, mask_path, args.show, args.model).mask()

def create_mask(image_path):
    pic_path = image_path
    mask_path = "/media/preeth/Data/prajna_files/mask_creator/face_mask/images/blue-mask.png"

```

```

show = False
model = "hog"
FaceMasker(pic_path, mask_path, show, model).mask()

class FaceMasker:
    KEY_FACIAL_FEATURES = ('nose_bridge', 'chin')

    def __init__(self, face_path, mask_path, show=False, model='hog'):
        self.face_path = face_path
        self.mask_path = mask_path
        self.show = show
        self.model = model
        self._face_img: ImageFile = None
        self._mask_img: ImageFile = None

    def mask(self):
        import face_recognition

        face_image_np = face_recognition.load_image_file(self.face_path)
        face_locations = face_recognition.face_locations(face_image_np, model=self.model)
        face_landmarks = face_recognition.face_landmarks(face_image_np, face_locations)
        self._face_img = Image.fromarray(face_image_np)
        self._mask_img = Image.open(self.mask_path)

        found_face = False
        for face_landmark in face_landmarks:
            # check whether facial features meet requirement
            skip = False
            for facial_feature in self.KEY_FACIAL_FEATURES:
                if facial_feature not in face_landmark:
                    skip = True
                    break
            if skip:
                continue

            # mask face
            found_face = True
            self._mask_face(face_landmark)

        if found_face:
            if self.show:
                self._face_img.show()

            # save
            self._save()
        else:
            print('Found no face.')

    def _mask_face(self, face_landmark: dict):
        nose_bridge = face_landmark['nose_bridge']

```

```

nose_point = nose_bridge[len(nose_bridge) * 1 // 4]
nose_v = np.array(nose_point)

chin = face_landmark['chin']
chin_len = len(chin)
chin_bottom_point = chin[chin_len // 2]
chin_bottom_v = np.array(chin_bottom_point)
chin_left_point = chin[chin_len // 8]
chin_right_point = chin[chin_len * 7 // 8]

# split mask and resize
width = self._mask_img.width
height = self._mask_img.height
width_ratio = 1.2
new_height = int(np.linalg.norm(nose_v - chin_bottom_v))

# left
mask_left_img = self._mask_img.crop((0, 0, width // 2, height))
mask_left_width = self.get_distance_from_point_to_line(chin_left_p
oint, nose_point, chin_bottom_point)
mask_left_width = int(mask_left_width * width_ratio)
mask_left_img = mask_left_img.resize((mask_left_width, new_height)
)

# right
mask_right_img = self._mask_img.crop((width // 2, 0, width, height
))
mask_right_width = self.get_distance_from_point_to_line(chin_right
_point, nose_point, chin_bottom_point)
mask_right_width = int(mask_right_width * width_ratio)
mask_right_img = mask_right_img.resize((mask_right_width, new_heig
ht))

# merge mask
size = (mask_left_img.width + mask_right_img.width, new_height)
mask_img = Image.new('RGBA', size)
mask_img.paste(mask_left_img, (0, 0), mask_left_img)
mask_img.paste(mask_right_img, (mask_left_img.width, 0), mask_righ
t_img)

# rotate mask
angle = np.arctan2(chin_bottom_point[1] - nose_point[1], chin_bott
om_point[0] - nose_point[0])
rotated_mask_img = mask_img.rotate(angle, expand=True)

# calculate mask location
center_x = (nose_point[0] + chin_bottom_point[0]) // 2
center_y = (nose_point[1] + chin_bottom_point[1]) // 2

offset = mask_img.width // 2 - mask_left_img.width
radian = angle * np.pi / 180
box_x = center_x + int(offset * np.cos(radian)) - rotated_mask_img
.width // 2

```

```

        box_y = center_y + int(offset * np.sin(radian)) - rotated_mask_img
        .height // 2

        # add mask
        self._face_img.paste(mask_img, (box_x, box_y), mask_img)

    def _save(self):
        path_splits = os.path.splitext(self.face_path)
        new_face_path = path_splits[0] + '-with-mask' + path_splits[1]
        self._face_img.save(new_face_path)
        print(f'Save to {new_face_path}')

    @staticmethod
    def get_distance_from_point_to_line(point, line_point1, line_point2):
        distance = np.abs((line_point2[1] - line_point1[1]) * point[0] +
                           (line_point1[0] - line_point2[0]) * point[1] +
                           (line_point2[0] - line_point1[0]) * line_point1[
1] +
                           (line_point1[1] - line_point2[1]) * line_point1[
0]) / \
            np.sqrt((line_point2[1] - line_point1[1]) * (line_point
2[1] - line_point1[1]) +
                    (line_point1[0] - line_point2[0]) * (line_point
1[0] - line_point2[0]))
        return int(distance)

if __name__ == '__main__':
    #cli()
    create_mask(image_path)

```