МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження та програмна реалізація алгоритмів обробки зображень на мобільних платформах

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 "Комп'ютерна інженерія"
(освітня програма - "Системне програмування")

Науковий керівник роботи:	_____ (підпис)	М.Є. Щербакова (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О. Критська (ініціали, прізвище)
Студент:	_____ (підпис)	К.М. Прядко (ініціали, прізвище)
Група:		СП-16дм

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки
(шифр і назва)
Спеціальність 123 - "Комп'ютерна інженерія"

(освітня програма - "Системне програмування")
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри
І.С. Скарга-Бандурова
«_____» _____20____р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Прядко Катерині Миколаївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та програмна реалізація алгоритмів обробки зображень на мобільних платформах
керівник проекту (роботи) Щербакова М.Є., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " 18 " 10 2017 року № 207/48
2. Строк подання студентом проекту (роботи) 18.01.2018
3. Вихідні дані до роботи Матеріали науково-дослідної практики, мова програмування – Java, C++, середовище розробки – Android Studio

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Огляд літератури.
Методи аналізу зображень.
Штучні нейронні мережі.
Практична реалізація.
Підготовка програмного забезпечення.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Критська Я.О.		
Основна частина	Щербакова М.Є.		

7. Дата видачі завдання 18.10.2017

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	*Аналіз завдання*	*05.09.17-10.09.17*	
2	*Літературний пошук. Визначення вимог до роботи.*	*11.09.17-16.10.17*	
3	*Огляд відомих засобів і методів рішень*	*17.10.17-30.10.17*	
4	*Дослідження і побудова рішення*	*01.11.17-18.11.17*	
5	*Практична реалізація*	*19.11.17-05.12.17*	
6	*Розробка заходів з охороні праці*	*06.12.17-18.12.17*	
7	*Оформлення пояснювальної записки і графічного матеріалу*	*19.12.17-25.12.17*	
8	*Підготовка та подання магістерської роботи до захисту*	*26.12.17-16.01.18*	

Студент _____ Прядко К.М.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Щербакова М.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Прядко К.М. Дослідження та програмна реалізація алгоритмів обробки зображень на мобільних платформах.

Метою атестаційної роботи є адаптація та прискорення роботи алгоритмів обробки зображень з використанням нейронних штучних мереж на мобільних платформах. Робота присвячена дослідженню застосування нейронних мереж в задачах виявлення образів та об'єктів на зображеннях та на вхідному відео потоці.

В атестаційній роботі проведено дослідження застосування нейронних мереж для виявлення та визначення коефіцієнту імовірності знаходження на зображенні того чи іншого об'єкту. В результаті дослідження був розроблений програмний додаток для мобільних пристроїв під управлінням операційної системи Android, що класифікує у реальному часі об'єкти, які знаходяться у фокусі камери пристрою.

Ключові слова: нейронна мережа, згорточна мережа, тенсор, граф, вейвлет, нейрочіп

АНОТАЦИЯ

Прядко Е.Н. Исследование и программная реализация алгоритмов обработки изображений на мобильных платформах.

Целью аттестационной работы является адаптация и ускорения работы алгоритмов обработки изображений с использованием нейронных искусственных сетей на мобильных платформах. Работа посвящена исследованию применения нейронных сетей в задачах выявления образов и объектов на изображениях и на входном видеопотоке.

В аттестационной работе проведено исследование применения нейронных сетей для выявления и определения коэффициента вероятности нахождения на изображении того или иного объекта. В результате исследования было разработано программное приложение для мобильных устройств под управлением операционной системы Android, классифицирующее в реальном времени объекты, которые находятся в фокусе камеры устройства.

Ключевые слова:нейронные сети, сверточная сеть, тенсор, граф, вейвлет, нейрочип

ABSTRACT

Pryadko K.N. Research and software realization of image processing algorithms on a mobile platforms.

The purpose of thesis is to adapt the work and speed up algorithms using neural networks on mobile platforms. The work deals with the application of neural networks in problems of identification of images and objects in the images and on the input video stream.

In attestation work studied the application of neural networks to identify and rate the probability of finding an image of an object. In result software application has been developed for mobile devices running the Android operating system, classifies objects in real time, that are in the device's camera focus.

Key words: neural network, convolutional network, tensor, graph, neurochip

ЗМІСТ

ВСТУП

Обґрунтування вибору теми досліджень. В останні кілька років спостерігається вибух інтересу до нейронних мереж, вони знаходять успішне застосування в самих різних областях - бізнесі, медицині, геології, фізики. Нейронні мережі увійшли в практику всюди, де потрібно вирішувати завдання прогнозування, класифікації та управління. Такі характеристики нейронних методів, як можливість нелінійного моделювання та простота реалізації, часто роблять їх незамінними при вирішенні найскладніших багатовимірних задач.

Нейронні мережі нелінійні за своєю природою і представляють виключно потужний метод моделювання, що дозволяє відтворювати надзвичайно складні залежності. Протягом багатьох років в якості основного методу в більшості областей використовувалося лінійне моделювання, оскільки для нього добре розроблені процедури оптимізації. Там, де лінійна апроксимація незадовільна і лінійні моделі працюють погано, а таких завдань досить багато, основним інструментом стають нейронні методи. Крім того, нейронні мережі справляються з «прокляттям розмірності», яке не дозволяє моделювати лінійні залежності в разі великого числа змінних [1].

Нейронні мережі навчаються на прикладах. Користувач нейронної мережі підбирає репрезентативну вибірку, а потім запускає алгоритм навчання, який автоматично сприймає структуру даних. При цьому від користувача, звичайно, потрібно якийсь набір евристичних знань про те, як слід відбирати і готувати дані, вибирати потрібну архітектуру мережі та інтерпретувати результати, проте рівень знань, необхідний для успішного застосування нейронних мереж, набагато скромніше, ніж, наприклад, при використанні традиційних методів статистики.

Нейронні мережі привабливі з інтуїтивної точки зору, бо вони засновані на біологічній моделі нервових систем. В майбутньому розвиток таких нейробіологічних моделей може привести до створення дійсно мислячих комп'ютерів, які зможуть давати прогнози на майбутнє в будь-якій області діяльності людини [2].

Основні завдання, які ставляться перед нейронними мережами, відносяться до завдань розпізнавання образів. Вони полягають в тому, щоб класифікувати вхідний образ, тобто віднести його до якого-небудь відомого мережі класу. Спочатку мережі даються еталонні образи - такі образи, приналежність яких до певного класу відома. Потім на вхід мережі подається деякий невідомий образ, і мережа намагається за певним алгоритмом співвідносити його з до будь-якого еталонного образу. Можна сказати, що нейромережі проводять кластеризацію образів

Все вищенаведене свідчить про те, що проблема дослідження можливостей нейронних мереж і їх розвитку є актуальною на даний момент.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне** завдання розробки нової технології, яка дає змогу генерувати опис процесу рішення задачі нейронною мережею.

Об'єктом дослідження магістерської роботи є прискорення роботи алгоритмів обробки зображень на мобільних пристроях.

Предметом дослідження є методи застосування нейронних мереж в задачах виявлення образів і об'єктів на зображеннях.

Мета і задачі дослідження. Метою роботи є дослідження можливості прискорення роботи алгоритмів обробки та аналізу графічних даних.

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

1. Розглянути існуючі роботи, засоби і методи та провести дослідження застосування нейронних мереж для виявлення та визначення коефіцієнту імовірності знаходження на зображенні того чи іншого об’єкту.

2. Розглянути основні складності застосування нейронних мереж, знайти оптимальні шляхи вирішення.

3. Розробити програмний додаток для мобільних пристроїв під управлінням операційної системи Android, що класифікує у реальному часі об’єкти, які знаходяться у фокусі камери пристрою.

Методи рішення поставлених задач базуються на комплексному використанні нейромережевих методів. Використовуючи таблицю експериментальних даних, що описують предметну область, можна буде отримати явний алгоритм вирішення поставленого завдання.

Наукова новизна одержаних результатів.

Одержав подальший розвиток методів застосування нейронних мереж для виявлення та визначення коефіцієнту імовірності знаходження на зображенні того чи іншого об’єкту.

Особистий внесок здобувача полягає у розроблені нової моделі програмного додатку для мобільних пристроїв під управлінням операційної системи Android для визначення об’єктів на зображенні.

Апробація результатів роботи. Основні результати магістерської атестаційної роботи докладалися на VII всеукраїнської науково-практичної конференції «Електронні апарати та системи. Проблеми створення. Перспективи розвитку», VIII всеукраїнської науково-практичної конференції «Майбутній науковець – 2017» (м. Сєвєродонецьк)**.**

Практичне значення отриманих результатів полягає у тому, що можливість запуску нейронної мережі на мобільному пристрої свідчить про швидкий розвиток і великі подальші

перспективи технології. Тепер не потрібно мати потужний обчислювальний центр, щоб користуватися перевагами НМ. І у майбутньому, нейронні мережі зможуть працювати швидше і точніше навіть на мобільному пристрої.

Публікації. Основні результати магістерської атестаційної роботи опубліковано в 2 наукових працях: серед яких тези на всеукраїнської науково-практичної конференції «Електронні апарати та системи. Проблеми створення. Перспективи розвитку», та тези доповіді на всеукраїнської науково-практичної конференції «Майбутній науковець – 2017»**.**

Структура і обсяг роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. Загальний обсяг роботи складає 127 сторінок, з яких анотація на 1 сторінці, зміст на 2 сторінках, вступ на 3 сторінках, основний текст на 86 сторінках, висновки на 1 сторінці, список використаних джерел із 40 найменувань на 3 сторінках, додатки на 31 сторінках. Робота містить 6 таблиць та 28 рисунків.

1 ОГЛЯД СИСТЕМ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

У розділі проведено аналіз систем розпізнавання зображення, проаналізовані основні галузі їх використання, особливості застосування. Поставлена наукова задача та обґрунтована методика досліджень.

1.1 Системи розпізнавання зображень

В даний час все більш широке поширення набувають біометричні системи ідентифікації особи. Традиційні системи ідентифікації вимагають знання пароля, наявності ключа, ідентифікаційної картки, або іншого предмета, що ідентифікує, який не можна забути або втратити. На відміну від них біометричні системи ґрунтуються на унікальних біологічних характеристиках людини, які важко підробити і які однозначно визначають конкретну людину. До таких характеристик відносяться відбитки пальців, форма долоні, візерунок райдужної оболонки, зображення сітківки ока [3].

Події останнього часу, особливо терористичний акт в Нью-Йорку, безумовно, істотно вплинули на світовий ринок біометрії. Так наприклад, поміж громадян США всього 10% підтримувало ідею біометричної паспортизації до 11 вересня 2001 року і вже понад 75% - після теракту, коли відстеження потенційно небезпечних особистостей стало першочерговим завданням. Біометрія отримала вагомий шанс стати однією з найбільш швидко поширюючихся галузей безпеки.

Сьогодні світовий ринок біометричних систем формують понад 300 компаній, які займаються розробкою, виробництвом, продажем і обслуговуванням засобів і систем безпеки. Структура цього ринку досить досконально була вивчена фахівцями фірми IDC і наведена на рис. 1.1 (станом на 2012 рік).

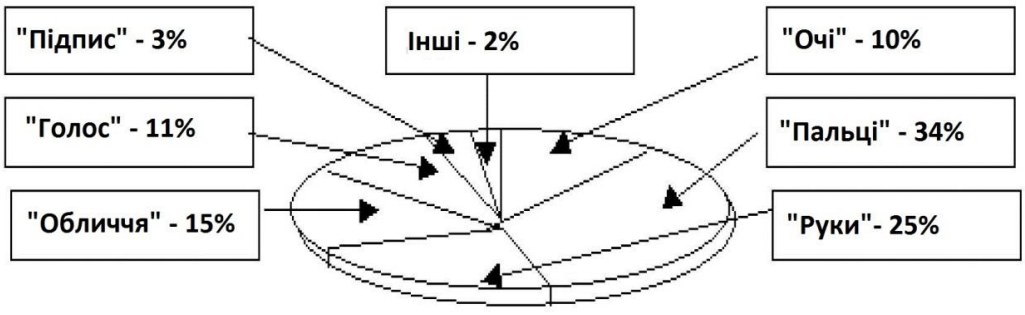


Рис. 1.1 - Структура світового ринку біометричних засобів захисту

Аналіз ринку чітко показує, що основний інтерес покупця складають технології та засоби контролю доступу в будівлі і до комп'ютерів. Середній річний темп розвитку біометрії становить 40%, що є високим показником навіть для зростаючої економіки. При збереженні таких темпів через 15 років кількість населення Землі буде забезпечено біометричними посвідченнями особи, інформація про які буде зберігатися в державних базах даних, об'єднаних в глобальну міжнародну ідентифікаційну систему.

В даний час вітчизняною промисловістю і рядом зарубіжних фірм пропонується досить широкий набір різних засобів контролю доступу до інформації, в результаті чого вибір оптимального їх поєднання для застосування в кожному конкретному випадку виростає в самостійну проблему. За своїм походженням на українському ринку в даний час представлені як вітчизняні, так і імпортні біометричні засоби захисту інформації (БЗЗІ), хоча існують і спільно розроблені засоби. За конструктивними особливостями можна відзначити системи, виконані у вигляді моноблока, кількох блоків і у вигляді приставок до комп'ютерів. Можлива класифікація біометричних засобів захисту інформації за біометричними ознаками, принципам дії і технологією реалізації приведена на рис. 1.2.

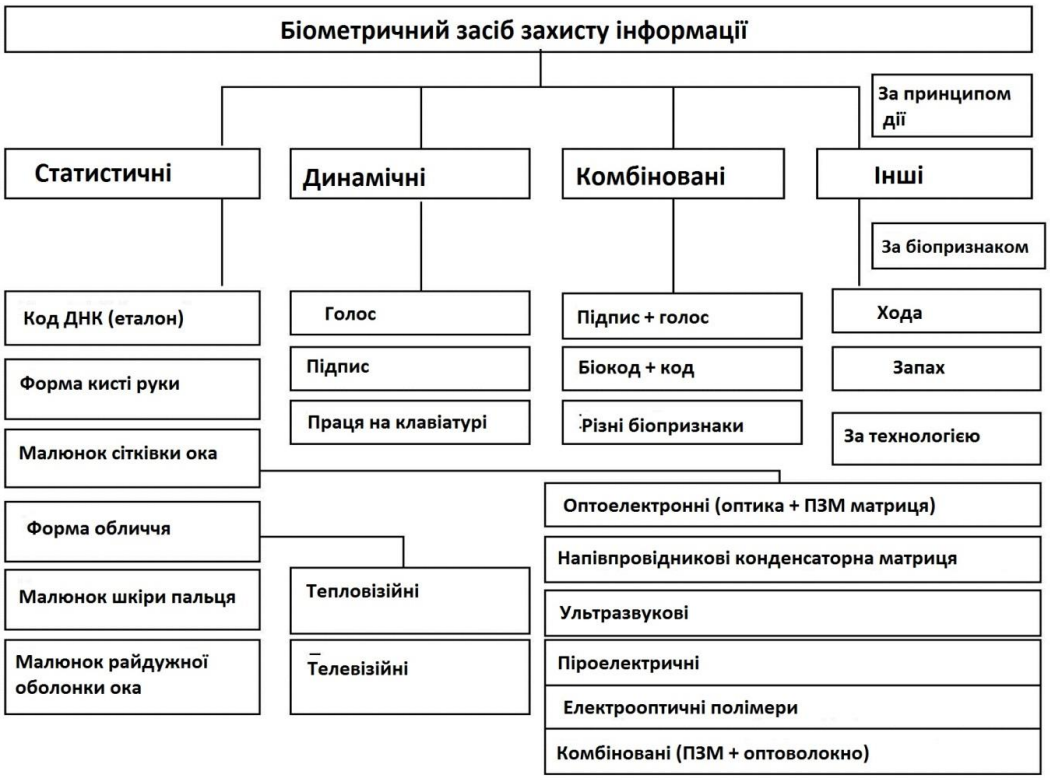


Рис. 1.2 - Класифікація біометричних засобів захисту інформації

В даний час біометричні системи контролю доступу до інформації завойовують все більшу популярність в банках, фірмах, пов'язаних із забезпеченням безпеки в телекомунікаційних мережах, в інформаційних відділах фірм і т. д. Розширення застосування

систем цього типу можна пояснити як зниженням їх вартості, так і підвищенням вимог до рівня безпеки. Подібні системи на російському ринку з'явилися завдяки фірмам "Identix", "SAC Technologies", "Eyedentify", "Biometric Identification Inc.", "Recognition Systems", "Trans-Ameritech", "BioLink", "Sonda", "Elsys" , "Едванс», «ААМ Системз", "Маском", "Біометричні системи" та ін.

Всі біометричні системи характеризуються високим рівнем безпеки, перш за все тому, що використовувані в них дані не можуть бути загублені користувачем, викрадені або скопійовані. В силу свого принципу дії багато біометричні системи поки ще відрізняються порівняно малою швидкодією і низькою пропускною здатністю. Тим не менш, вони представляють собою єдине рішення проблеми контролю доступу на особливо важливих об'єктах. Наприклад, біометрична система може контролювати доступ до інформації і сховищ в банках, її можна використовувати на підприємствах, зайнятих обробкою цінної інформації, для захисту RGB, засобів зв'язку і т.д. За оцінками фахівців, більше 85% встановлених в США засобів біометричного контролю доступу призначалися для захисту машинних залів ЕОМ, сховищ цінної інформації.

Розпізнавання людини по зображенню особи виділяється серед біометричних систем тим, що по-перше, не потрібно спеціальне або дороге устаткування. Для більшості додатків досить персонального комп'ютера і звичайної відеокамери. По-друге, не потрібен фізичний контакт з пристроями. Не треба ні до чого торкатися або спеціально зупинятися й чекати спрацьовування системи. У більшості випадків достатньо просто пройти повз або затриматися перед камерою на невеликий час [5].

Основні програми біометричної ідентифікації особистості по зображенню особи у різних сферах людської діяльності наведені в таблиці 3.1.

Основна відмінність даних додатків між собою цільові класи, які є об'єктами розпізнавання. Цільовими класами в задачі розпізнавання особи можуть бути: особа, особа з елементами перекриттів, міміка обличчя, стать, раса, особистість людини. Вибір одного з таких цільових класів і визначає специфіку алгоритму розпізнавання, інші класи є другорядними і швидше грають роль ознак при розпізнаванні цільового класу.

До недоліків розпізнавання людини по зображенню особи слід віднести те, що сама по собі така система не забезпечує 100% -ої надійності ідентифікації [14]. Там, де потрібна висока надійність, застосовують комбінування декількох біометричних методів (мультимодальні біометричні системи) [5].

На даний момент проблеми розпізнавання людини по зображенню особи присвячено безліч робіт, однак в цілому вона ще далека від вирішення. Основні труднощі полягають у

тому, щоб розпізнати людину по зображенню особи незалежно від зміни ракурсу та умов освітлення при зйомці, а так само при різних змінах, пов'язаних з віком, зачіскою і т.д.

Розпізнавання зображень пересікається з розпізнаванням образів. Такі завдання не мають точної аналітичного рішення. При цьому потрібно виділення ключових ознак, що характеризують зоровий образ, визначення відносної важливості ознак шляхом вибору їх вагових коефіцієнтів і облік взаємозв'язків між ознаками. Спочатку ці завдання виконувалися людиною-експертом вручну, шляхом експериментів, що займало багато часу і не гарантувало якості. У нових методах виділення ключових ознак здійснюється шляхом автоматичного аналізу навчальної вибірки, але тим не менше більшість інформації про ознаки задається вручну. Для автоматичного застосування таких аналізаторів вибірка повинна бути досить великою і охоплювати всі можливі ситуації.

Таблиця 1.1 - Застосування біометричної ідентифікації в людській діяльності

Сфера діяльності	Додатки
Інформаційна безпека	Безпека доступу (ОС, бази даних), збереження особистих даних (медичних, тощо), аутентифікація користувача (торгівля, он-лайн банкінг)
Контроль доступу	Аутентифікація в системах Безпеки. Системи пов'язані з закритим доступом.
Біометрика	Ідентифікація особистості (паспорта, водійські права, посвідчення голосуючого і т.п.), автоматичне підтвердження особистості (митний контроль)
Правоохоронні органи	Відеоспостереження Ідентифікація підозрюваних Відстеження осіб, яких підозрюють в процесі розслідувань Реконструкція осіб за свідченнями свідків
Особиста безпека	Домашні системи відеоспостереження. Системи контролю стану людини (наприклад, система анти-сон для водіїв)
Розваги і відпочинок	Системи інтерактивних відеоігор. Додатки фото і відео камер

Нейромережеві методи пропонують інший підхід до вирішення задачі розпізнавання образів [11]. Архітектура і функціонування нейронних мереж (НМ) мають біологічні прообрази. Ваги в нейронної мережі не обчислюються шляхом вирішення аналітичних рівнянь, а підлаштовуються різними локальними методами (наприклад різновидами градієнтного спуску) при навчанні. Навчаються нейронні мережі на наборі навчальних прикладів. У процесі навчання НМ відбувається автоматичне вилучення ключових ознак, визначення їх важливості і побудова взаємозв'язків між ними. Навчена НМ може успішно застосовувати досвід, отриманий в процесі навчання, на невідомі образи за рахунок хороших узагальнюючих здібностей.

При всьому різноманітті різних алгоритмів і методів розпізнавання зображень вони мають схожу структуру. Типовий метод розпізнавання складається з (рис. 1.3):

1. перетворення вихідного зображення в початкове уявлення (може включати в себе як предобробку, так і математичні перетворення, наприклад обчислення основних компонентів);
2. виділення ключових характеристик (наприклад, беруться перші n головних компонент або коефіцієнтів дискретного косинусного перетворення);
3. механізм класифікації (моделювання): кластерна модель, метрика, нейронна мережа і т.п.;

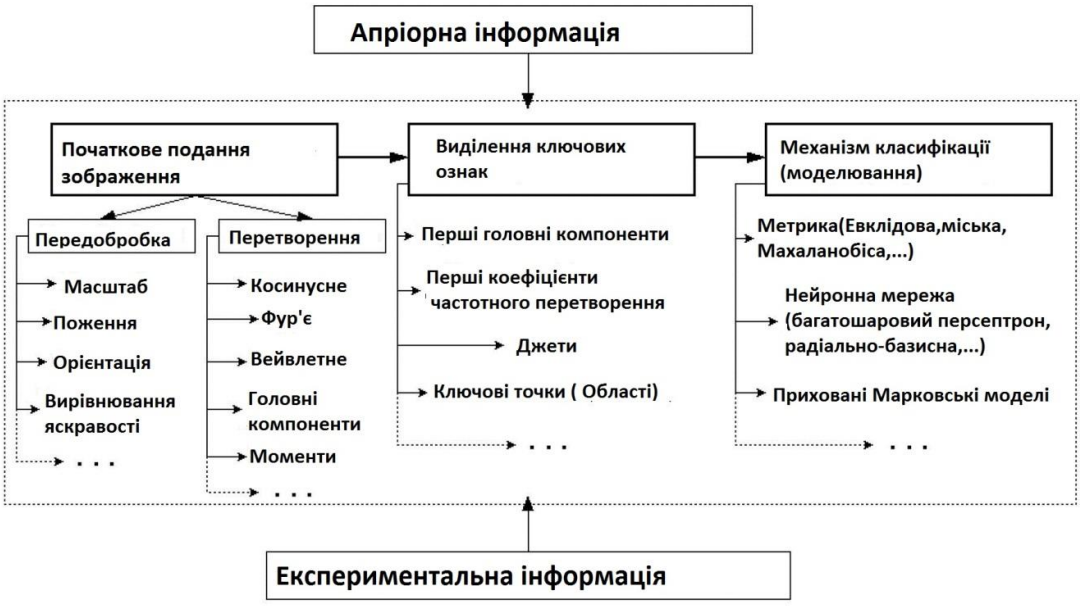


Рис. 1.3 - Структура методу розпізнавання зображень

Крім цього, побудова методу розпізнавання спирається на апріорну інформацію про предметну область (в даному випадку - характеристики особи людини), і коригується експериментальною інформацією, що з'являється по ходу розробки методу.

1.2 Особливості застосування нейронних мереж

Перші роботи, що відносяться до моделювання штучних нейронних мереж, з'явилися понад чотири десятки років тому. Ці дослідження представлялися вельми перспективними завдяки дивовижним властивостям, притаманним штучним нейронним мережам, що нагадує роботу мозку (здатність до узагальнення інформації, до вилучення істотних властивостей з даних, до навчання і самонавчання на основі власного досвіду функціонування і т.п.). Однак незабаром настала пора розчарувань і затишшя, можливо через недостатньо розвинену в той час напівпровідникову технологію і перемикання уваги дослідників на нові моделі і методи обробки та використання знань.

Після багатьох років майже повного забуття інтерес до методів нейромережевої обробки інформації виник знову, зокрема, в зв'язку з розвитком мікроелектронної технології і розробкою принципово нових фізичних принципів реалізації елементів і фрагментів нейронних мереж, в тому числі мереж з дуже великим числом нейронів. Це в свою чергу різко активізувало дослідження в області нейроматематики (методів вирішення завдань в нейромережевому базисі) і в області архітектур нового типу - нейрокомп'ютерів. Основна відмінність нейрокомпьютерів від традиційних обчислювальних машин полягає в тому, що основою обчислень є не алгоритм в класичному його розумінні і поданні (наприклад, у вигляді граф схеми алгоритму або у вигляді логічної схеми алгоритму), а якийсь спосіб обчислень в нейромережевому логічному базисі [2].

Саме ця особливість і дозволяє розглядати нейромережеву обробку інформації в якості окремого напрямку сучасної інформаційної технології.

Перш за все, коли говоримо про нейронні мережи, то частіше маємо на увазі штучні нейронні мережі. Деякі штучні нейронні мережі моделюють біологічні нейронні мережі, деякі ні. Однак історично склалося так, що перші штучні нейронні мережі, були створені в результаті спроб отримати комп'ютерну модель, що відтворює діяльність мозку в спрощеній формі. Звичайно, можливості людського мозку незмірно більше, ніж можливості найпотужнішої штучної нейронної мережі. Однак штучні нейромережі мають ряд властивостей властивих біологічним нейромереж, в тому числі і людському мозку.

Головна властивість нейромереж - здатність до навчання. Для вирішення будь-якої задачі на комп'ютері традиційним методом необхідно знати правила (математичні формули), за якими з вхідних даних можна отримає вихідні (знайти рішення задачі). За допомогою нейромережі можна знайти рішення, не знаючи правил, а маючи кілька прикладів. Нейромережі використовують підхід до вирішення завдань ближчий до людського, ніж традиційні

обчислення. Справді, наприклад, коли людина переходить вулицю, вона оцінює швидкість руху автомобіля виходячи з попереднього досвіду не використовуючи математичних обчислень. Або, наприклад, як дитина без праці може відрізнити кішку від собаки, грунтуючись на раніше бачених їм прикладах. При цьому часто вона не може точно сказати, за якими ознаками їх відрізняє, тобто не знає чіткого алгоритму.

Інша важлива властивість нейромереж - здатність знаходити рішення, ґрунтуючись на перекручених і навіть суперечливих даних.

Ще одна чудова властивість - відмовостійкість. У разі виходу з ладу частини нейронів вся мережа в цілому продовжує залишатися працездатною, хоча, звичайно, точність знижується. Це властивість важлива для апаратно реалізованих нейромереж, тому що, якщо нейромережа імітується на традиційному комп'ютері, то в разі виходу з ладу центрального процесора вся нейромережа втратить працездатність [3].

В принципі нейронні мережі можуть обчислити будь-яку функцію, що має рішення. Іншими словами, робити все, що можуть робити традиційні комп'ютери.

На практиці, для того, щоб застосування нейронної мережі було виправдано, необхідно, щоб задача мала наступні ознаки:

1. відсутній алгоритм або не відомі принципи вирішення завдань, але накопичена достатня кількість прикладів;
2. проблема характеризується великими обсягами вхідної інформації;
3. дані неповні або надлишкові, зашумлені, частково суперечливі.

Таким чином, нейромережі добре підходять для розпізнавання образів і вирішення завдань класифікації, оптимізації і прогнозування. Нижче наведено перелік можливих промислових застосувань нейронних мереж, на базі яких або вже створені комерційні продукти, або реалізовані демонстраційні прототипи.

Банки і страхові компанії: автоматичне зчитування чеків і фінансових документів, перевірка достовірності підписів, прогнозування змін економічних показників.

Адміністративне обслуговування: автоматичне зчитування документів, автоматичне розпізнавання штрихових кодів.

Нафтова і хімічна промисловість: аналіз геологічної інформації, розвідка покладів мінералів за даними аерофотозйомок, аналіз складів домішок, управління процесами.

Військова промисловість і аеронавтика: обробка звукових сигналів (поділ, ідентифікація, локалізація, усунення шуму, інтерпретація), обробка радарних сигналів (розпізнавання цілей, ідентифікація і локалізація джерел), обробка інфрачервоних сигналів (локалізація), автоматичне пілотування.

Промислове виробництво: управління маніпуляторами, управління якістю, управління процесами, виявлення несправності, адаптивна робототехніка.

Служба безпеки: розпізнавання осіб, голосів, відбитків пальців.

Медицина: виявлення та ідентифікація ракових клітин, діагностування та прогнозування ймовірності виникнення захворювань, виявлення відхилень в ЕКГ, аналіз рентгенограм.

Телебачення і зв'язок: адаптивне управління мережею зв'язку, стиснення і відновлення зображення.

Представлений перелік далеко не повний.

Зрозуміло, зовсім не любе завдання можна вирішити за допомогою нейронної мережі. Якщо треба визначити результати лотереї, тираж якої відбудеться через тиждень, знаючи свій розмір взуття, то навряд чи це вийде, оскільки ці показники ніяк не пов'язані один з одним. Насправді, якщо тираж проводиться чесно, то не існує такої інформації, на підставі якої можна було б передбачити результат.

З вище сказаного можна виділити дві основні умови, коли можна застосовувати нейронні мережі:

1. Повинна бути конкретна відома інформація для навчання нейронної мережі.
2. Потрібен зв'язок між відомими вхідними і невідомими вихідними значеннями. Цей зв'язок може бути спотворений шумом, але він повинен існувати.

1.3 Архітектура нейронних мереж

Прототипом для створення цих елементів послужив біологічний нейрон. В основу штучних нейронних мереж покладено такі риси біологічних нейронних мереж, що дозволяють їм добре справлятися з нерегулярними завданнями:

- простий елемент, що обробляє - нейрон;
- дуже велика кількість нейронів бере участь в обробці інформації;
- один нейрон пов'язаний з великим числом інших нейронів (глобальні зв'язку);
- змінюються за вагою зв'язку між нейронами;
- масована паралельність обробки інформації.

Спрощено, можна вважати, що нейрон влаштований і діє таким чином. Біологічний нейрон має тіло, сукупність відростків - дендритів, за якими в нейрон надходять сигнали, і відросток - аксон, що передає вихідні сигнали іншим нейронам. Точка з'єднання дендрита і аксона називається синапсом. Синапс виконує функції вагового коефіцієнта, посилюючи або послаблюючи вхідний сигнал. Нейрон отримує від дендритів набір вхідних сигналів. У тілі нейрона значення вхідних сигналів підсумовується. У штучному нейроні обчислюється

скалярний добуток вектора вхідних сигналів і вектора вагових коефіцієнтів. Потім нейрон формує вихідний сигнал, інтенсивність якого залежить від значення обчисленого скалярного добутку. Вихідний сигнал надходить на аксон, а через нього передається дендритам інших нейронів.

Розвиток штучних нейронних мереж надихається біологією. Тобто, розглядаючи мережеві конфігурації і алгоритми, дослідники застосовують терміни, запозичені з принципів організації мозкової діяльності. Але на цьому аналогія закінчується. Наші знання про роботу мозку настільки обмежені, що мало б знайшлося точно доведених закономірностей для тих, хто побажав би керуватися ними. Тому розробникам мереж доводиться виходити за межі сучасних біологічних знань в пошуках структур, здатних виконувати корисні функції. У багатьох випадках це призводить до необхідності відмови від біологічної правдоподібності, мозок стає просто метафорою, і створюються мережі, неможливі в живій матерії або можуть бути використані неправдоподібно великі допущення про анатомію і функціонування мозку.

Незважаючи на те, що зв'язок з біологією слабкий і часто несуттєвий, штучні нейронні мережі продовжують порівнювати з мозком. Їх функціонування часто має зовнішню схожість з людським пізнанням, тому важко уникнути цієї аналогії. На жаль, такі порівняння неплідни і створюють невиправдані очікування, що неминуче ведуть до розчарування.

Нервова система людини, побудована з елементів, званих нейронами, має приголомшуючу складність. Близько10^{11} нейронів беруть участь в приблизно передавальних зв'язках, що мають довжину метр і більше. Кожен нейрон має багато властивостей, спільними з іншими органами тіла, але йому притаманні абсолютно унікальні здібності: приймати, обробляти і передавати електрохімічні сигнали по нервових шляхах, які утворюють комунікаційну систему мозку.

На рис. 1.4 показано будову «нейронної мережі» людини. Дендрити йдуть від тіла нервової клітини до інших нейронів, де вони приймають сигнали в точках з'єднання, які називаються синапсами. Прийняті синапсом вхідні сигнали передаються до тіла нейрона. Тут вони підсумовуються, причому одні входи прагнуть порушити нейрон, інші - перешкодити його порушенню.

Коли сумарне збудження в тілі нейрона перевищує деякий поріг, нейрон збуджується, посилаючи по аксону сигнал іншим нейронам. У цієї основної функціональної схеми багато ускладнень і винятків, проте, більшість штучних нейронних мереж моделюють лише ці прості властивості.

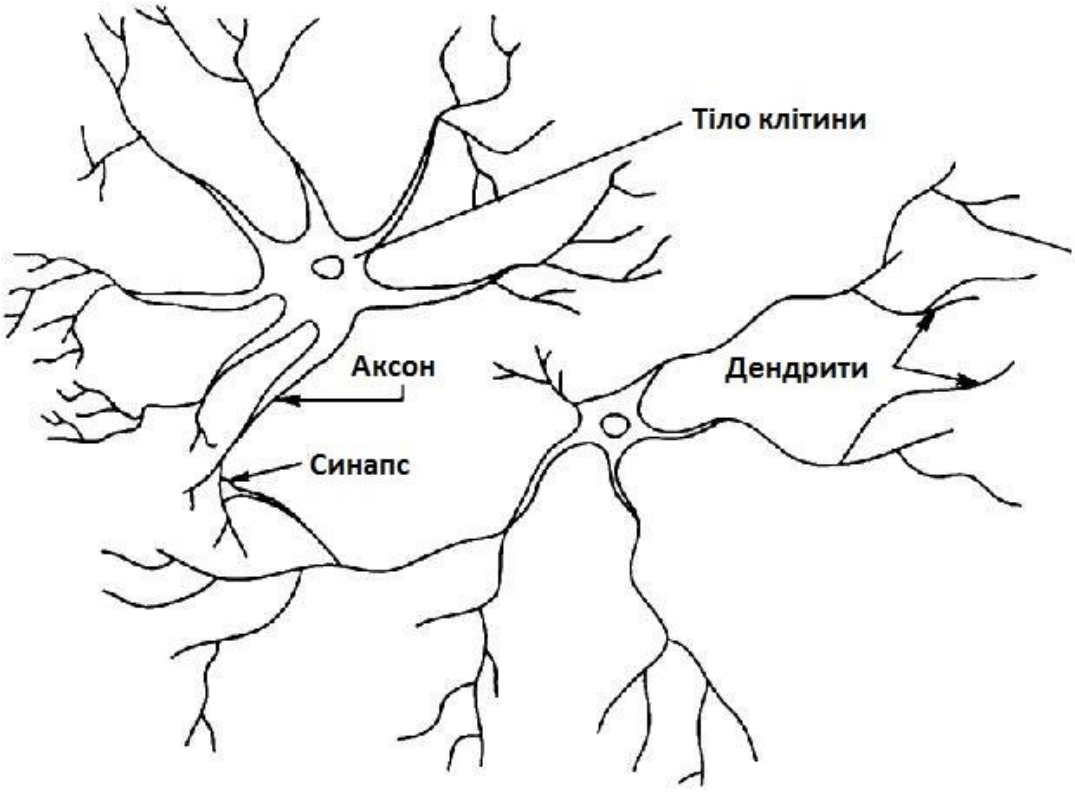


Рис. 1.4 - Типові біологічні нейрони

Штучний нейрон імітує в першому наближенні властивості біологічного нейрона. На вхід штучного нейрона надходить деяка безліч сигналів, кожен з яких є виходом іншого нейрона. Кожен вхід множиться на відповідну вагу, аналогічний синаптичній силі, і всі добутки підсумовуються, визначаючи рівень активації нейрона [2].

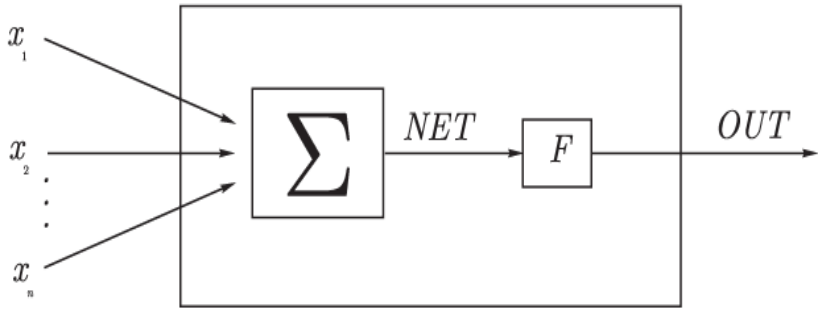


Рис. 1.5 - Штучний нейрон

На рис. 1.5 представлена модель, що реалізує цю ідею. Безліч вхідних сигналів, позначених $x_1, x_2, \ldots, x_n$, надходить на штучний нейрон. Ці вхідні сигнали в сукупності позначаються вектором X, відповідають сигналам, що приходять в синапси біологічного нейрона. Кожен сигнал множиться на відповідну вагу $w_1, w_2, \ldots, w_n$, і надходить на сумуючий

блок, позначений $\sum$. Кожна вага відповідає "силі" одного біологічного синаптичного зв'язку. Сумуючий блок, що відповідає тілу біологічного елемента, складає зважені входи алгебраїчно, створюючи вихід, який будемо називати NET. У векторних позначеннях це може бути записане таким чином:

$$NET = XW \tag{1.1}$$

Сигнал NET далі, як правило, перетворюється активаційною функцією F і дає вихідний нейронний сигнал OUT. Активаційна функція може бути звичайною лінійною функцією

$$OUT = F(NET) \tag{1.2}$$

де F - гранична функція.

$$OUT = \begin{cases} 1, \text{если } NET > T; \\ 0, \text{если } NET \leq T; \end{cases} \tag{1.3}$$

де T - деяка постійна порогова величина.

На рис.1.2 блок, позначений F, приймає сигнал NET і видає сигнал OUT. Якщо блок F звужує діапазон зміни величини NET так, що при будь-яких значеннях NET значення OUT належать деякому кінцевому інтервалу, то F називається функцією, що "стискає". Як "стискуючи" функції часто використовується логістична або "сигмоїдальна" (S-образна) функція, показана на рис.1.6.

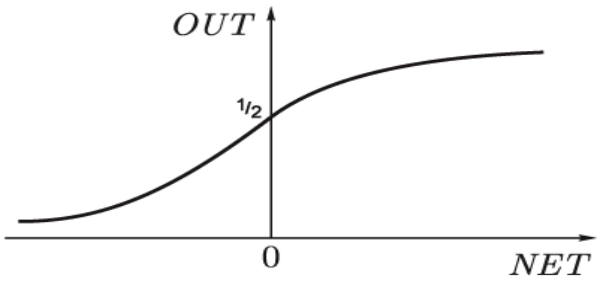


Рис. 1.6 - Функція F

Ця функція математично виражається як

$$F(x) = \frac{1}{(1+ e^{-x})}. \quad (1.4)$$

Таким чином,

$$OUT = \frac{1}{1 + e^{-NET}} \quad (1.5)$$

За аналогією з електронними системами активаційну функцію можна вважати нелінійною підсилювальною характеристикою штучного нейрона. Коефіцієнт посилення обчислюється як відношення приросту величини OUT до викликав його невеликому збільшенню величини NET. Він виражається нахилом кривої при певному рівні збудження і змінюється від малих значень при великих негативних збудженнях (крива майже горизонтальна) до максимального значення при нульовому збудженні і знову зменшується, коли збудження стає великим позитивним. С. Гросберг (1973) виявив, що подібна нелінійна характеристика вирішує поставлену їм дилему шумового насичення. Яким чином одна і та ж мережа може обробляти як слабкі, так і сильні сигнали? Слабкі сигнали мають потребу у великому мережевому посиленні, щоб дати придатний до використання вихідний сигнал. Однак підсилювальні каскади з великими коефіцієнтами посилення можуть привести до насичення виходу шумами підсилювачів (випадковими флуктуаціями), які присутні в будь-якій фізично реалізованій мережі. Сильні вхідні сигнали, в свою чергу, також будуть приводити до насичення підсилювальних каскадів, виключаючи можливість корисного використання виходу. Центральна область логістичної функції, що має великий коефіцієнт посилення, вирішує проблему обробки слабких сигналів, в той час як області з падаючим посиленням на позитивному і негативному кінцях підходять для великих збуджень. Таким чином, нейрон функціонує з великим посиленням в широкому діапазоні рівня вхідного сигналу.

$$OUT = \frac{1}{1 + e^{-NET}} = F(NET) \quad (1.6)$$

Іншою широко застосуємо активаційною функцією є гіперболічний тангенс. За формою він схожий з логістичною функцією і часто використовується біологами як математична модель активації нервової клітини. Як активаційна функція штучної нейронної мережі вона записується в такий спосіб

$$OUT=th(x) \quad (1.7)$$

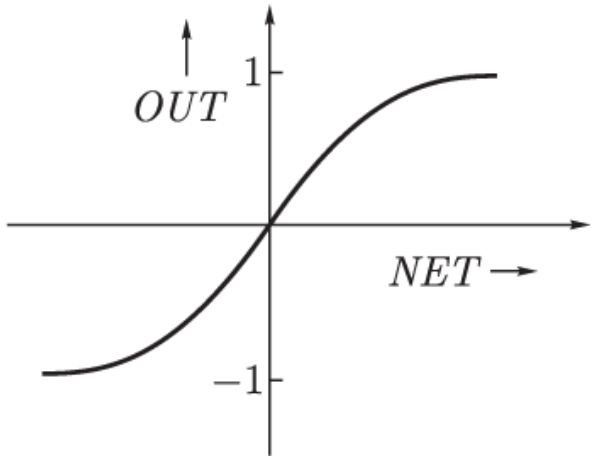


Рис.1. 7 - Функція F

Подібно логістичній функції гіперболічний тангенс є S-образною функцією, але він симетричний відносно початку координат, і в точці NET = 0 значення вихідного сигналу OUT дорівнює нулю (рис. 1.7). На відміну від логістичної функції, гіперболічний тангенс приймає значення різних знаків, і ця його властивість застосовується для цілого ряду мереж.

Розглянута проста модель штучного нейрона ігнорує багато властивостей свого біологічного двійника. Наприклад, вона не бере до уваги затримки в часі, які впливають на динаміку системи. Вхідні сигнали відразу ж породжують вихідний сигнал. І, що більш важливо, вона не враховує впливів функції частотної модуляції або синхронізуючої функції біологічного нейрона, які ряд дослідників вважають вирішальними в нервовій діяльності природного мозку.

Незважаючи на ці обмеження, мережі, побудовані з таких нейронів, виявляють властивості, які сильно нагадують біологічну систему. Тільки час і дослідження зможуть відповісти на питання, чи є подібні збіги випадковими або ж вони є наслідком того, що в моделі вірно схоплені найважливіші риси біологічного нейрона.

Нейронні мережі можуть мати зворотний зв'язок (тобто зв'язки від виходів деяких нейронів до входів інших нейронів), а можуть не мати їх. Мережі Хопфілда - це нейронні мережі із зворотними зв'язками, причому вихід кожного нейрона зв'язується зі входами всіх інших нейронів. Так як мережі з зворотними зв'язками мають шляхи, що передають сигнали від виходів до входів, то відгук таких мереж є динамічним, тобто після додавання нового входу обчислюється вихід і, передаючись по мережі зворотного зв'язку, модифікує вхід. Потім вихід повторно обчислюється, і процес повторюється знову і знову. Для стійкої мережі послідовні ітерації приводять до все менших змін виходу, поки врешті-решт вихід не стає постійним. Для багатьох мереж процес ніколи не закінчується, такі мережі називають нестійкими. Нестійкі мережі мають цікаві властивості і вивчалися як приклад хаотичних систем. Ніхто не міг

передбачити, які з мереж будуть стійкими, а які будуть перебувати в постійній зміні. Більш того, проблема представлялася настільки важкою, що багато дослідників були налаштовані песимістично щодо можливості її рішення [4].

На щастя, була отримана теорема, що описала підмножину мереж із зворотними зв'язками, виходи яких зрештою досягають стійкого стану. Це чудове досягнення відкрило дорогу подальшим дослідженням, і сьогодні багато вчених займаються дослідженням складної поведінки і можливостей цих систем. Дж. Хопфілд зробив важливий внесок як в теорію, так і в застосування систем із зворотними зв'язками. Тому деякі з конфігурацій відомі як мережі Хопфілда. Зупинимося на важливому окремому випадку нейромережевої архітектури, для якої властивості стійкості детально досліджені. На рис. 1.8. показана мережа із зворотними зв'язками, що складається з двох шарів. Спосіб подання дещо відрізняється від використаного в роботі Хопфілда і інших подібних, але еквівалентний їм з функціональної точки зору. Нульовий шар не виконує обчислювальної функції, а лише розподіляє виходи мережі назад на входи. Кожен нейрон першого шару обчислює зважену суму своїх входів, даючи сигнал NET, який потім за допомогою нелінійної функції F перетворюється в сигнал OUT. Ці операції схожі з нейронами інших мереж.

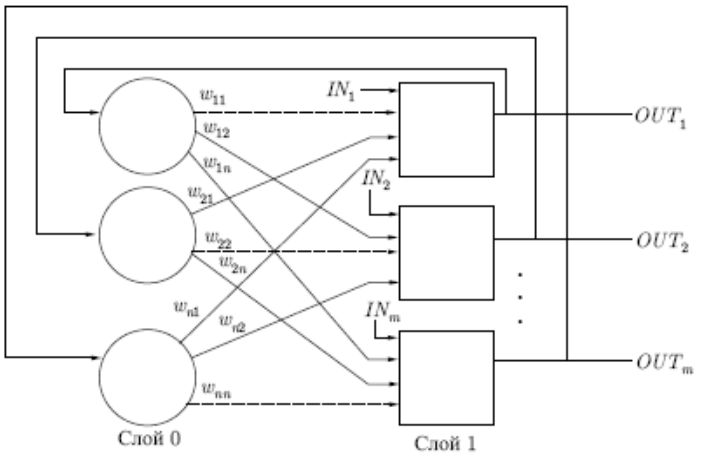


Рис. 1.8 - Мережа Хопфілда

Стан мережі - це просто безліч поточних значень сигналів OUT від всіх нейронів. У початковій мережі Хопфілда стан кожного нейрона змінювалося в дискретні випадкові моменти часу, в подальшому - стани нейронів могли змінюватися одночасно. Так як виходом бінарного нейрона може бути тільки нуль або одиниця (проміжних рівнів немає), то поточний стан мережі є двійковим числом, кожний біт якого є сигналом OUT деякого нейрона [3].

Завдання, які вирішуються цією мережею, як правило, формуються наступним чином. Відомий деякий набір двійкових сигналів (зображень, оцифровок звуку, інших даних, що

описують якісь об'єкти або характеристики процесів), які вважаються зразковими. Мережа повинна вміти з довільного неідеального сигналу, поданого на її вхід, виділити ("згадати" по частковій інформації) відповідний зразок (якщо такий є) або "дати висновок" про те, що вхідні дані не відповідають жодному із зразків. У загальному випадку, будь-який сигнал може бути описаний вектором

$$X = \{x_i : i = 0 \dots n-1\} \tag{1.8}$$

n- число нейронів в мережі і розмірність вхідних і вихідних векторів. Кожен елемент дорівнює або 1, або 0. Позначимо вектор, що описує k-й зразок, через X_k, де k = 0 ... m - 1, m- число зразків. Коли мережа розпізнає (або "згадає") будь-який зразок на основі пред'явлених їй даних, її виходи будуть містити

$$Y = X^k \tag{1.9}$$

де Y - вектор вихідних значень мережі

$$Y = \{y_i : i = 0 \dots n-1\} \tag{1.10}$$

В іншому випадку, вихідний вектор не зійдеться з одним зразковим.

Якщо, наприклад, сигнали являють собою якісь зображення, то, відобразивши в графічному вигляді дані з виходу мережі, можна буде побачити картинку, яка повністю збігається з однією зі зразкових (у разі успіху) або ж "вільну імпровізацію" мережі (у разі невдачі).

На стадії ініціалізації мережі вагові коефіцієнти синапсів встановлюються таким чином

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, i = j \end{cases} \tag{1.11}$$

Тут і та j - індекси, відповідно, предсинаптичного і постсинаптичного нейронів; x_i^k, x_j^k — i-й и j-й елементи вектора k-го зразка.

Алгоритм функціонування мережі наступний (p- номер ітерації):

1. На входи мережі подається невідомий сигнал. Фактично його введення здійснюється безпосередньою установкою значень аксонів:

$$y_i(0) = x_i, i = 0, \dots, n-1, \tag{1.12}$$

тому позначення на схемі мережі вхідних синапсів в явному вигляді носить чисто умовний характер. Нуль в дужках означає нульову ітерацію в циклі роботи мережі.

2. Розраховується новий стан нейронів:

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), j = 0, \dots, n-1 \tag{1.13}$$

і нові значення аксонів:

$$y_j(p+1) = f\left(s_j(p+1)\right) \tag{1.14}$$

де f - активаційна функція.

3. Перевірка, чи змінилися вихідні значення аксонів за останню ітерацію. Якщо так - перехід до пункту 2, інакше (якщо виходи стабілізувались) - кінець процедури. При цьому вихідний вектор являє собою зразок, який найкращим чином поєднується з вхідними даними.

1.3.1 Формування нейронної мережі

Для вирішення різних практичних завдань потрібні різні моделі нейронних мереж. Модель нейронної мережі визначається моделями нейронів і структурою зв'язків мережі.

Залежно від структури зв'язків можна виділити кілька груп нейронних мереж:

1. Багатошарові нейронні мережі. Нейрони в таких мережах діляться на групи із загальним вхідним сигналом - шаром.

2. Повнозв'язні нейронні мережі. Кожен нейрон в повнозв'язних мережах пов'язаний з усіма іншими. На кожному такті функціонування мережі на входи нейронів подається вхідний сигнал і виходи нейронів попереднього такту.

3. Нейронні мережі з локальними зв'язками. Нейрони в таких мережах розташовуються у вузлах прямокутної решітки. Кожен нейрон пов'язаний з невеликим числом (4 або 8) своїх топологічних сусідів.

4. Неструктуровані нейронні мережі. До цієї групи належать всі моделі нейронних мереж, які не можна віднести ні до однієї з попередніх груп.

Моделі нейронів, які використовуються в нейронних мережах, надзвичайно різноманітні. У найпростішому випадку нейрон складається з умножителей (синапсів), суматора і нелінійного перетворювача. Нейрон виконує скалярную функцію векторного аргументу - зважене підсумовування компонент вектора вхідного сигналу і нелінійне перетворення результату підсумовування. Такий нейрон називається нейроном першого порядку. Нейрони більш високих порядків здійснюють множення двовимірних матриць і багатовимірних тензорів.

У моделях нейронів використовується безліч різних варіантів нелінійних перетворювачів. Функція, що реалізується нелінійним перетворювачем, називається функцією активації або передавальною функцією нейрона. Найбільш часто використовуються сигмоїдальні, кусочно-лінійні і жорсткі порогові функції активації.

Якщо в мережі все нейрони мають однакові функції активації, то мережу називається однорідною (гомогенною). У неоднорідних (гетерогенних) мережах нейрони мають різні функції активації.

Для побудови нейронної мережі, орієнтованої на вирішення конкретного завдання, використовуються процедури формування (чи створення) нейронних мереж. Ці процедури забезпечують введення зазначених характеристик моделей нейронів і структур нейронних мереж. Як правило, в кожній окремій програмі реалізована лише частина з описаних моделей нейронів і нейронних мереж [2].

1.3.2 Навчання нейронної мережі

Для того щоб нейронна мережа придбала здатність вирішувати конкретну задачу, тобто на кожен вхідний сигнал видавати необхідний вихідний сигнал, необхідно провести налаштування параметрів мережі. Налаштування проводиться за навчальною вибіркою, яка складається з пар («вхід»,« бажаний вихід>) - навчальних прикладів.

Залежно від розв'язуваної задачі в навчальній вибірці використовуються ті чи інші типи даних і різні розмірності вхідних / вихідних сигналів. Вхідні дані прикладів навчальної вибірки - зображення, таблиці чисел. Типи вхідних даних - бінарні (0 і 1), біполярні (-1 і 1) числа, цілі чи дійсні числа з деякого діапазону. Вихідні сигнали мережі - вектора цілих або дійсних чисел. Для вирішення практичних завдань часто потрібні навчальні вибірки великого обсягу. Через жорстко обмежений обсяг оперативної пам'яті комп'ютера розмістити в ній великі навчальні вибірки неможливо. Тому вибірка ділиться на сторінки - групи прикладів. У кожен момент часу лише одна сторінка прикладів розташовується в пам'яті комп'ютера, інші - на жорсткому диску. Сторінки послідовно завантажуються в пам'ять комп'ютера.

В даний час відсутня універсальна методика побудови навчальних вибірок. Набір навчальних прикладів формується за бажанням користувача програми моделювання нейронних мереж індивідуально для кожної конкретної розв'язуваної задачі.

Якщо в ненавчену нейронну мережу ввести вхідний сигнал одного з прикладів навчальної вибірки, то вихідний сигнал мережі буде істотно відрізнятися від бажаного вихідного сигналу, визначеного в навчальній вибірці. Функція помилки чисельно визначає подібність всіх поточних вихідних сигналів мережі і відповідних бажаних вихідних сигналів навчальної вибірки. Найбільш поширеною функцією помилки є середньоквадратичне відхилення. Однак запропоновані й інші функції помилки. Головна мета навчання - мінімізувати функцію помилки, тобто знайти такі значення параметрів мережі, при яких поточні вихідні сигнали мережі мінімально відрізняються від відповідних бажаних вихідних сигналів, заданих навчальною вибіркою.

Для навчання нейронних мереж можуть бути використані різні алгоритми. Можна виділити дві великі групи алгоритмів - градієнтні і стохастичні. Градієнтні алгоритми навчання мереж засновані на обчисленні приватних похідних функції помилки за параметрами мережі. Серед градієнтних розрізняють алгоритми першого і другого порядків. В стохастичних алгоритмах пошук мінімуму функції помилки ведеться випадковим чином. При навчанні мереж, як правило, використовується один з двох наступних критеріїв зупинки: зупинка при досягненні деякого мінімального значення функції помилки або зупинки в разі успішного вирішення всіх прикладів навчальної вибірки [4]. Перед навчанням виконується ініціалізація нейронної мережі, тобто присвоювання параметрам мережі деяких початкових значень. Як правило, ці початкові значення - деякі малі випадкові числа.

Для формування навчальних вибірок, ініціалізації і навчання в програмах моделювання нейронних мереж використовуються спеціальні процедури. Можливість використання багатосторінкового навчання є дуже важливою при вирішенні практичних завдань за допомогою нейронних мереж, що моделюються на звичайних комп'ютерах. Навчання - це ітераційна процедура, яка при реалізації на звичайних комп'ютерах, вимагає значного часу.

1.3.3 Імітація функціонування (тестування) навченої нейронної мережі

Для перевірки навичок, набутих мережею в процесі навчання, використовується імітація функціонування мережі. У мережу вводиться деякий сигнал, який, як правило, не збігається ні з одним з вхідних сигналів прикладів навчальної вибірки. Далі аналізується вихідний сигнал, що вийшов з мережі. Тестування навченої мережі може проводитися на одиночних вхідних сигналах, або на контрольній вибірці, яка має структуру, аналогічну навчальній вибірці, і також

складається з пар («вхід», «бажаний вихід»). Як правило, навчальна і контрольна вибірки не перетинаються. Контрольна вибірка будується користувачем індивідуально для кожної розв'язуваної задачі.

1.4 Постановка наукової задачі та обґрунтування методики досліджень

В результаті проведеного аналізу моделей, було виявлено істотний недолік: найчастіше використовуються моделі створені на мовах високого рівня під кожен конкретний експеримент, що приводить до невиправдано високих витрат і низької швидкодії систем.

В цьому контексті можна виділити наступні задачі:

— розробка нового додатку, що дасть змогу використовувати нейромережу не маючи потужного обчислювального центру;

— проведення експерименту для оцінки якості роботи метода.

Для проведення досліджень доцільно застосовувати методи, які б допомогли зменшити витрати ресурсу системи.

1.5 Висновки до розділу

У розділі було показано існуючі алгоритми обробки зображень. Після огляду існуючих методів та засобів було виділено нейромережеву технологію. Було розглянуто особливості її застосування та її архітектура. Для вирішення поставленого завдання треба мати на увазі:

Нейронна мережа використовується тоді, коли невідомий точний вид зв'язків між входами і виходами, і вона знаходить цю залежність в процесі навчання.

Кожна група моделей нейронних мереж може бути використана для вирішення лише деякого обмеженого класу практичних задач. Так багатошарові і повно зв'язкові нейронні мережі з сигмоїдальними передавальними функціями використовуються для розпізнавання образів і адаптивного управління; нейронні мережі з локальними зв'язками - для обробки зображень і деяких інших приватних завдань. Для вирішення завдань лінійної алгебри використовуються багатошарові мережі з особливими передавальними функціями.

Алгоритми навчання істотно розрізняються за швидкістю збіжності. Однією з найважливіших характеристик програм для моделювання нейронних мереж є швидкість збіжності алгоритму (або алгоритмів) навчання, які реалізовані в програмі.

Для імітації функціонування в переважній більшості програм моделювання нейронних мереж реалізовані спеціальні процедури.

2 ВИБІР МЕТОДУ АНАЛІЗУ ЗОБРАЖЕНЬ

У розділі детально розглянут та проаналізованоі існуючі методи розпізнавання образів з використанням нейромереж, було проведено аналіз архітектур кожного методу та обрано найбільш відповідну для вирішення завдання. Також були проаналізовані методи оптимізації нейронних мереж, проведено порівняння між існуючими аналогами.

2.1 Штучні нейронні мережі.

Завдання розпізнавання людини по зображенню особи діляться на три великі класи: пошук в великих базах даних, контроль доступу та контроль фотографій в документах. Вони розрізняються як за вимогами, що надаються до систем розпізнавання, так і щодо способів вирішення, і тому являють собою окремі класи.

Різні і вимоги, що пред'являються до помилок першого і другого роду для таких класів [5]. Помилкою першого роду (type I error, misdetection) називається ситуація, коли об'єкт заданого класу не розпізнається (пропускається) системою. Помилка другого роду (type II error, false alarm) відбувається, коли об'єкт заданого класу приймається за об'єкт іншого класу. Слід так само відзначити відмінність понять верифікації та розпізнавання (ідентифікації) [14]. В задачі верифікації невідомий об'єкт заявляє, що він належить до деякого відомому системі класу. Система підтверджує або спростовує цю заяву. У системах верифікації помилкою першого така є ситуація, коли об'єкт, що належить до відомих системі класів, приймається за об'єкт, що відноситься до невідомих системі класів, і в доступі йому відмовляють [13]. Помилка другого роду відбувається, коли об'єкт невідомого класу приймається за об'єкт, що відноситься до відомих системі класів, і йому дозволяється доступ [13]. При розпізнаванні потрібно віднести об'єкт до одного з відомих класів або видати висновок про те, що цей об'єкт не належить до відомих класів.

Порівняння типу «один з багатьма». Високі вимоги до помилки першого роду - система розпізнавання повинна знаходити зображення, відповідні даній людині, по можливості не пропустивши жодного такого зображення. При цьому допустимо, якщо в результуючій вибіркі буде присутнє невелике число інших людей.

Зазвичай у великій базі даних (104-107 зображень) потрібно знайти зображення, найбільш схожі на задане. Пошук повинен бути проведений за розумний час. Одне з рішень полягає в зберіганні в базі даних невеликих наборів заздалегідь витягнутих ключових ознак, які максимально характеризують зображення. При цьому вимоги до точності не настільки

критичні, як у задачах контролю доступу та документного контролю. До даного класу насамперед відноситься метод головних компонент (метод «власних осіб»). Коефіцієнти, отримані розкладанням вхідного зображення на головні компоненти, використовувалися для порівняння зображень шляхом обчислення Евклидової відстані, а в більш досконалих методах - на основі метрики Махала- Нобіса з використанням гауссовского розподілу.

На рис. 2.1 показаний алгоритм роботи системи при пошуку інформації в базі даних. Система спостереження робить фотографію особи. За допомогою нейронної мережі проводиться пошук області обличчя на цій фотографії. Область особи виділяється, оптимізуються яскравість, контраст зображення, потім нормалізований фотопортрет надходить на обробку другій нейронній мережі для розпізнавання. Нейронна мережа робе розпізнавання вхідного портрета і здійснює вибір з кількох найбільш схожих на нього портретів, які зберігаються у базі даних.

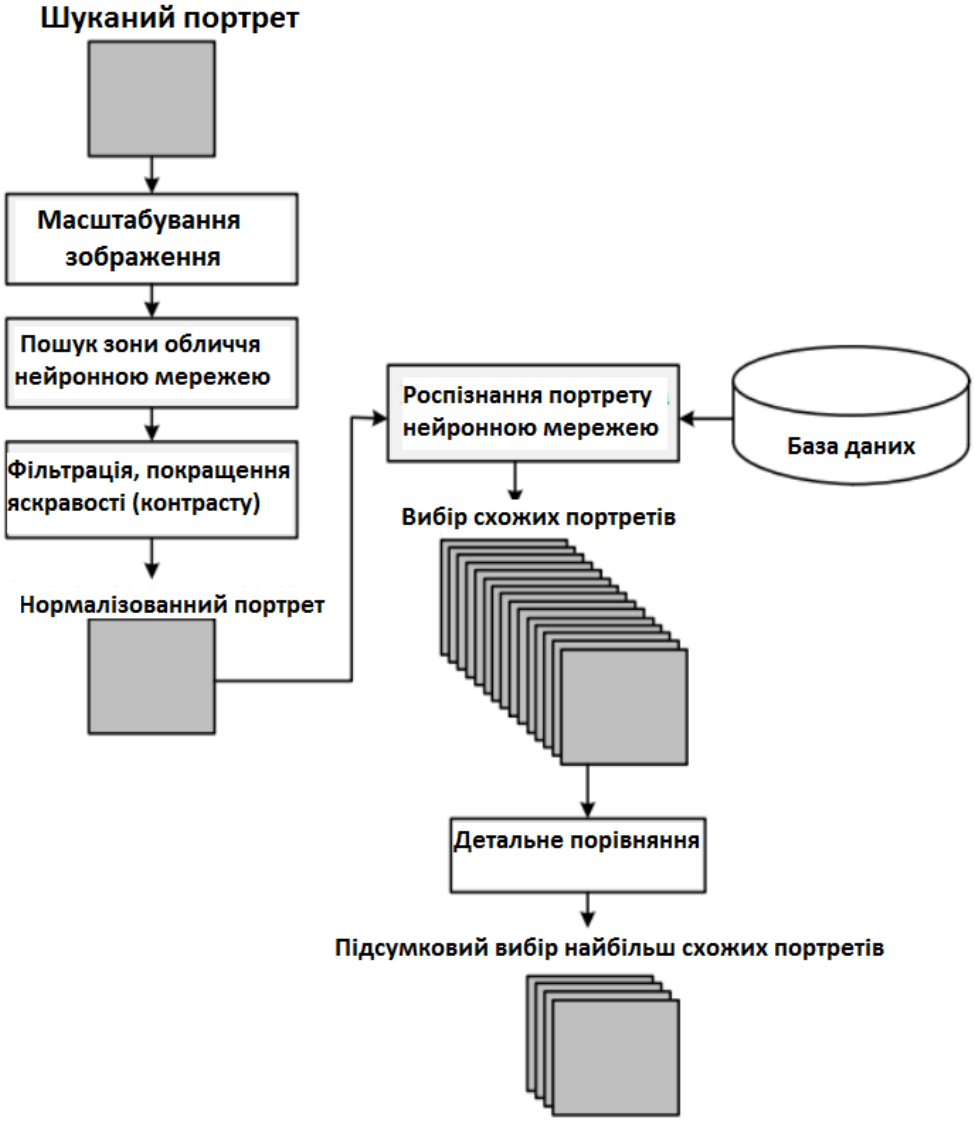


Рис. 2.1 - Алгоритм роботи системи пошуку

Порівняння типу «один з декількома». Критичними є вимоги до помилок другого роду. Система розпізнавання не повинна розпізнавати незнайомих людей як знайомих, можливо навіть за рахунок збільшення помилок першого роду (відмов у доступі знайомим людям).

Є невелика група осіб (5-50 чоловік), яких система повинна розпізнавати по зображенню особи і відкривати їм доступ в деякі приміщення. Людей, які не входять в цю групу, система не повинна пропускати. Можливі варіанти, коли потрібно встановити конкретну особу по зображенню особи. При цьому від системи потрібна висока достовірність розпізнавання, можливо навіть за рахунок збільшення числа відмов на знайомі об'єкти. Як тренувальних зображень зазвичай для кожної людини доступні кілька зображень особи, отриманих при різних умовах. Це можуть бути, наприклад, зміни ракурсу, умов освітленості, зачіски, міміки, наявність або відсутність окулярів, тощо. Система повинна працювати в реальному масштабі часу, а процес налаштування може займати багато часу і виконуватися попередньо. В процесі експлуатації система повинна до навчатися на тих зображеннях, що влаштовуються якомога швидше. Обмежень на застосовувані методу тут немає, але всі методи сходяться в тому, що є навчальний набір зображень осіб заданої групи людей (можливо при різних умовах зйомки). До цього набору система звертається в процесі розпізнавання або налаштовується на нього в процесі навчання.

Роботу подібної системи контролю доступу можна описати таким чином. На двері розташована фото- або відеокамера, яка фіксує людину на вході. Робиться фотознімок особи. На фотознімку знаходиться область розташування особи, далі відбувається розпізнавання цієї особи. Якщо особа відповідає портрету, що зберігається в базі даних, то читається додаткова інформація: ім'я, вік, посада і т. д. На основі цих даних система відкриває або закриває доступ до об'єктів. На рис. 2.2. показаний алгоритм роботи системи при введенні інформації в базу даних. На вхід надходить зображення, що представляє собою фотографію людини. За допомогою нейронної мережі проводиться пошук області обличчя на фотографії. Область особи виділяється, оптимізуються яскравість, контраст зображення, після чого даний фотопортрет зберігається в окремому файлі бази даних.

Порівняння типу «один з одним». Формулювати вимоги до помилок першого і другого роду як до системи верифікації або розпізнавання тут буде некоректно, оскільки система розпізнавання ніколи не мала справу з класами, що поступають на вхід. Але бажано, щоб система не скоювала помилок при порівнянні. Потрібно порівняти зображення обличчя людини, отримане в даний момент, з фотографією з будь-якого документа. Системі треба відповісти, чи належать ці особи одній людині чи ні. Даний клас завдань найбільш складний, оскільки, по-перше, система ніколи раніше не стикалася із зображенням обличчя даної людини.

Система порівнює зображення, які завжди відрізняються. Облік всіх можливих відмінностей в процесі навчання або настройки системи скрутні. По-друге, тут великий вплив мають вікові та інші зміни особи. По-третє, більшість методів для даного класу задач незастосовні без спеціальної адаптації. У літературі, що оглядається, немає робіт, безпосередньо пов'язаних із застосуванням нейромережевих методів для вирішення даного класу задач. Для цього можна запропонувати застосування НМ для вилучення ключових ознак зображень і адаптацію НМ для порівняння двох зображень.

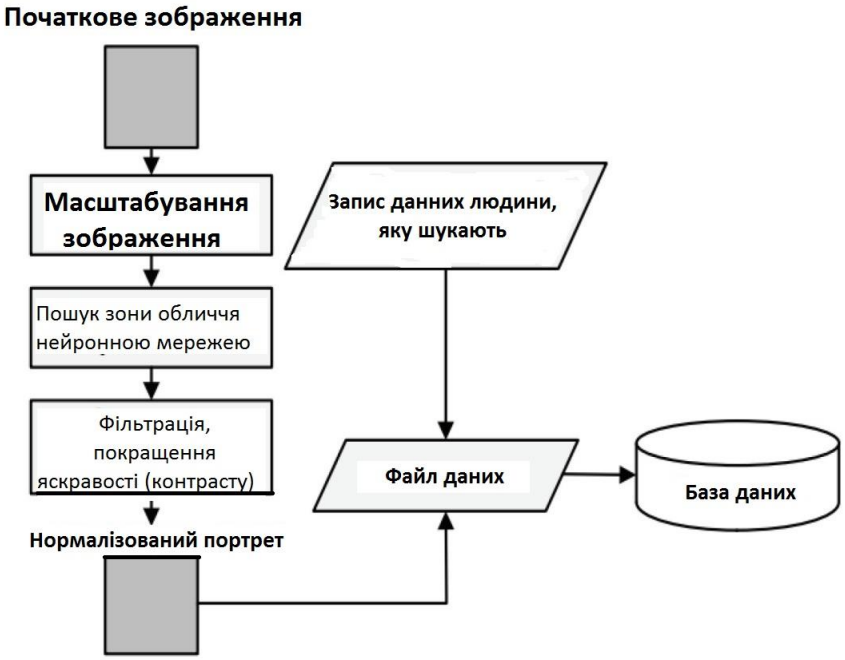


Рис. 2.2 - Алгоритм введення даних в систему контролю доступу.

2.1.1 Нейромережеві методи визначення по зображенню обличчя

Нейромережеві методи, засновані на застосуванні різних типів штучних нейронних мереж (ШНМ, надалі просто нейронні мережі, НМ), останнім часом набули широкого поширення. Основні задачі, які вирішуються за допомогою нейронних мереж наведені на рис.2.3.

Більшість з цих задач прямо або побічно пов'язані з розпізнаванням зображень. Основні переваги, якими володіють нейронні мережі, такі: налаштування нейронної мережі для вирішення певної задачі проводиться в процесі навчання на наборі тренувальних прикладів. Таким чином, не потрібно вручну визначати параметри моделі (вибирати ключові ознаки, враховувати їх взаємовідношення і т.п.) - НМ витягує параметри моделі автоматично найкращим чином в процесі навчання. Залишається тільки побудувати тренувальну вибірку. У

задачах класифікації при цьому відбувається неявне виділення ключових ознак всередині мережі, визначення значимості ознак і системи взаємовідносин між ними. В даний час розроблені потужні, гнучкі і універсальні механізми навчання різних типів НМ [2]. Крім того, архітектура НМ і процедури навчання дозволяють виконати гнучке налаштування на конкретну задачу. Для більшості НМ процедура навчання є евристичним алгоритмом, що, з одного боку, забезпечує прийнятність одержуваних рішень, а з іншого боку, не вимагає непомірних обчислювальних ресурсів.

Рис. 2.3 - Основні задачі, які вирішуються за допомогою нейронних мереж

Нейронні мережі мають гарну узагальнюючу здатністю (одну з кращих серед існуючих методів, наприклад, багато кращої, ніж в вирішальних дерев [11]). Це означає, що досвід, отриманий в процесі навчання на кінцевому наборі образів, НМ може успішно поширювати на всю безліч образів. Крім інтерполяційних узагальнюючих здатностей, НМ (багатошарові персептрони, наприклад) можуть добре екстраполювати, тобто застосовувати якісно свій досвід на інші образи, ніж ті, які зустрічалися в узагальнюючій вибірці.

Нейронні мережі ні накладають жодних обмежень на тренувальну вибірку, ні покладаються на те, що вона володіє будь-якими апріорними властивостями, на відміну, наприклад, від статистичних методів. Не потрібно ніякого попереднього вивчення характеру

даних. НМ приймає тренувальний набір «як є» і вчиться виробляти правдоподібне рішення, не претендуючи на абсолютну істину, тобто будується найкраща не фізична модель [12], яка не є максимально точною відповідністю реального процесу, але дає прийнятну його апроксимацію. Є ряд прикладів, коли нейронні мережі показували себе краще статистичних методів [12]. Крім того, в статистиці немає аналогів деяких нейромережевих методів [12], таких, наприклад, як карти Кохонена, машина Больцмана і, що важливо для розпізнавання зображень, когнітрон.

Природним чином архітектура НМ реалізується на паралельних обчислювальних засобах: спеціалізованих мікросхемах, оптичних і квантових комп'ютерах. Це відкриває широкі перспективи застосування НМ в майбутньому. НМ характеризується нечіткими розподіленим зберіганням інформації, тобто немає окремого нейрона, що відповідає за будь-яке поняття або ознаку, і видалення або спотворення роботи цього нейрона не призведе до фатальних наслідків.

Але незважаючи на всі переваги, застосування НМ до зображень вимагає спеціальних зусиль. Це пов'язано в першу чергу зі складним характером зображень, особливо зображень тривимірних об'єктів реального світу, якими і є обличчя людей. Зображення повинно бути предоброблено - приведено до деяких стандартних умов. Крім того, вибір початкового представлення зображення (це можуть бути, наприклад, частотні коефіцієнти, головні компоненти, вейвлетного коефіцієнти, моменти і т.п.) є окремою великою темою. Двовимірний характер зображення, зміна умов освітленості, топологічні спотворення зображення при зміні ракурсу та інших впливах не дозволяють обмежитися найпростішими архітектурою НМ для досягнення оптимального результату.

2.1.2 Класифікація нейронних мереж

В даний час крім багатошарового персептрона існує безліч способів завдання структур нейронних мереж. Всі види нейронних мереж можна умовно розділити на мережі прямого поширення і мережі з зворотними зв'язками. Як випливає з назви, в мережах першого типу сигнали від нейрона до нейрону поширюються в чітко заданому напрямку - від входів мережі до її виходів. У мережах другого типу вихідні значення будь-якого нейрона мережі можуть передаватися до його ж входів. Це дозволяє нейронній мережі моделювати складніші процеси, наприклад тимчасові, але робить виходи подібної мережі нестабільними, залежними від стану мережі на попередньому циклі. На рис. 2.4 представлена така класифікація найбільш найпоширеніших типів НМ.

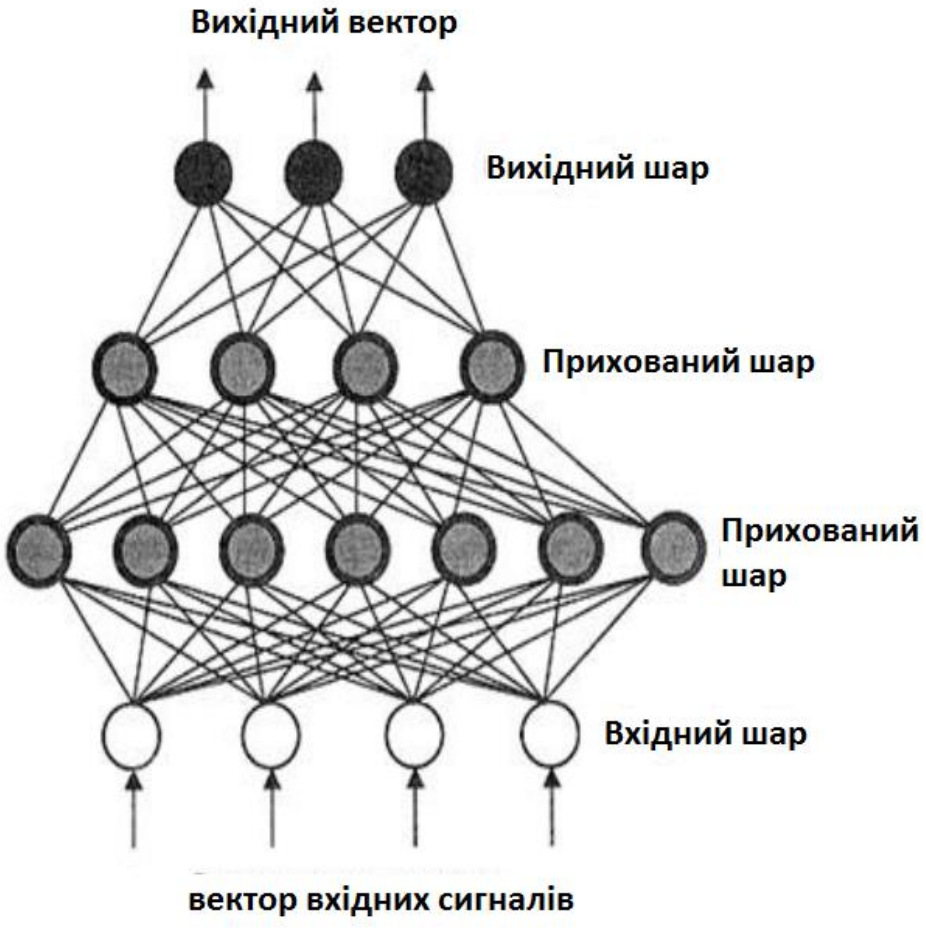


Рис. 2.4 - Схема багатошарового персептрона.

За характером зв'язків нейронні мережі можуть бути повнозв'язними, коли кожен нейрон пов'язаний з усіма іншими, і шаруватими, коли нейрони наступного шару пов'язані тільки з усіма нейронами попереднього шару. Ці дві архітектури є базовими, але можливі і різні варіації.

За характером функціонування нейронні мережі можуть бути однопрохідними, коли вихід мережі розраховується за один прохід мережі і релаксаційним, коли функціонування мережі триває до досягнення стабільного стану, це стан і є результатом роботи.

За характером формування зв'язків нейронні мережі можуть бути наступних видів:

1. Навчання з учителем: зв'язки настроюються в процесі навчання, причому еталонні значення результатів роботи відомі.

2. Самонавчання (навчання без вчителя): еталонні результати невідомі (не потрібні), мережа в процесі навчання повинна організувати вхідні образи на основі їх подібності.

3. Фіксовані зв'язки, які визначаються характером розв'язуваної задачі (наприклад в оптимізаційних завданнях).

Так само нейронні мережі можуть відрізнятися типом вхідної інформації (двійкова, аналогова і т.п.) і методом навчання.

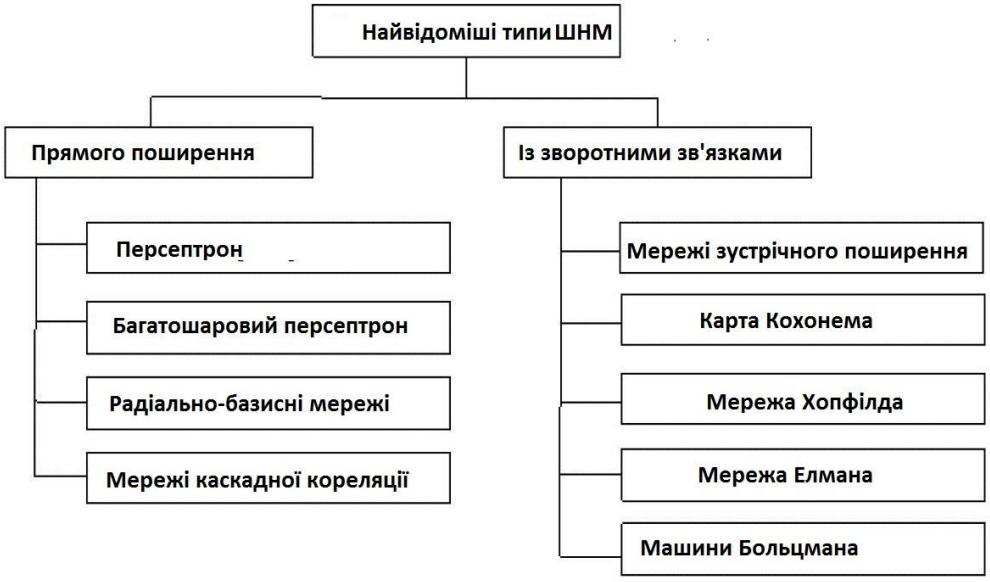


Рис. 2.5 - Класифікація поширених видів ШНМ.

Для вирішення цих завдань призначені багатошарові нейронні мережі, нейронні мережі високого порядку і радіально-базисні нейронні мережі. Оскільки такі мережі оперують у вихідному просторі зображення (ознак), то для них є критичним вимога предобробки зображення. Це приведення зображення до стандартного вигляду (положення, масштаб, орієнтація, вирівнювання яскравості), зниження розмірності даних, вибір ключових характеристик. Наступним наслідком оперування в вихідному просторі є неможливість обліку спотворення зображення (наприклад, при зміні ракурсу, емоцій), і тому тренувальна вибірка повинна містити репрезентативний набір прикладів, що представляють собою набори зображень об'єктів в тому діапазоні ракурсів і умов освітлення, в яких планується застосування системи розпізнавання [13].

2.1.3 Багатошарові нейронні мережі

Архітектура багатошарової нейронної мережі (БНМ, інша назва- багатошаровий персептрон, по-англійськи Multilayer Perceptron, MLP) складається з послідовно з'єднаних шарів, де нейрон кожного шару своїми входами пов'язаний з усіма нейронами попереднього шару, а виходами - наступного. Активаційними функціями для таких нейронів служать різновиди лінійних, порогових та сігмоідної функцій [6].

НМ з одним вирішальним шаром здатна формувати лінійні поверхні, що сильно звужує коло вирішуваних завдань, зокрема, така мережа не зможе вирішити завдання типу «виключає або». НМ з нелінійної функцією активації і двома вирішальними шарами дозволяє формувати

будь-які опуклі області в просторі рішень, а з трьома вирішальними шарами - області будь-якої складності в тому числі й неопуклої форми [7]. При цьому БНМ не втрачає своєї узагальнюючої здатності. За допомогою двошарової НМ можна з будь-якою точністю апроксимувати будь-яку багатовимірну функцію на відрізку від 0 до1. Навчаються БНМ за допомогою алгоритму зворотного поширення помилки, що є різновидом градієнтного спуску в просторі ваг з метою мінімізації сумарної помилки мережі:

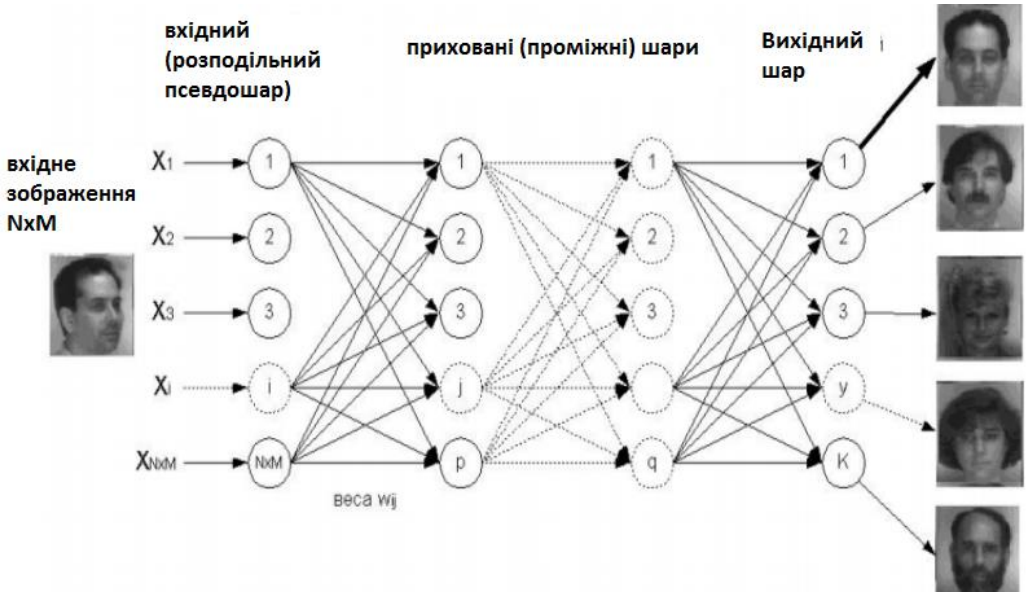


Рис. 2.6 - Архітектура багатошарової нейронної мережі і її застосування для розпізнавання зображень.

Нейрон з максимальною активністю (тут перший) вказує приналежність до розпізнаного класу

$$\Delta W = -\alpha \frac{dE}{dW}, \; E = \frac{1}{2}\sum_j (y_j - t_j)^2 \tag{2.1}$$

де t_j - еталонне значення виходів мережі. При цьому помилки (точніше величини корекції ваг) поширюються в зворотному напрямку від входів до виходів, крізь ваги, що з'єднують нейрони. Алгоритм зворотного поширення помилок є NP- важким, тому час навчання мережі збільшується експоненціально з ростом розмірності даних. Так як еталонні значення виходів відомі, алгоритм є навчанням з учителем. Але в застосуванні до вилучення ключових ознак, коли рециркуляційна мережа навчається реконструювати подане на вхід зображення, а на прихованих нейронах формується його стисле уявлення, алгоритм навчання може бути названий і самонавчанням. БНМ, як і більшість інших типів НМ, перед початком навчання

ініціалізуються випадковими вагами. Тому дві різні навчені НМ, що мають однаковий показник помилки, часто представлені абсолютно різними поверхнями [8].

Суть його полягає в тому, що є набір (колектив) мереж, навчених вирішувати одну і ту ж задачу, але різними способами (різна початкова ініціалізація ваг, архітектура, порядок проходження прикладів при навчанні і т.п.). Узагальнене рішення такого колективу як правило точніше (і надійніше), ніж рішення єдиної нейронної мережі. Існують так само методики синтезу колективу мереж, які виробляють максимально незалежні помилки [9].

Інший напрямок розвитку архітектури БНМ - це нейронні дерева [11,14]. В цій архітектурі вузлами вирішального дерева є нейронні мережі. У міру просування від кореня дерева мережі-вузли уточнюють рішення задачі. У порівнянні з такими методами, як звичайні нейронні дерева, вирішальні дерева, колективи НМ і багатошарові нейронні мережі, точність розпізнавання у запропонованого алгоритму була порівнянна або вище, а швидкість навчання і роботи - на порядок вище.

Одною з головних проблем БНМ (і деяких інших типів НМ) є наступні:

1. Проблема локального мінімуму. Як і для всіх градієнтних методів, проблема локального мінімуму полягає в тому, що при ітераційному спуску може наступити момент, коли рішення заходить в локальний мінімум, з якого внаслідок малої величини кроку не може вибратися. І такий локальний мінімум не завжди забезпечує прийнятне рішення. Вихід полягає в застосуванні стохастичних методів.

2. Вибір архітектури мережі (кількість нейронів, шарів, характер зв'язків). З цим також пов'язана проблема перенавчання, яка полягає в тому, що мережа з надмірною кількістю елементів втрачає узагальнюючу здатність і добре працює тільки на тренувальній вибірці. В даний час розроблені різні апріорні оцінки вибору архітектури, методи проріджування навчених мереж, методи «зростаючих» мереж.

3. Вибір кроку (швидкості) навчання. Така проблема пов'язана з тим, що при малому кроці час навчання буде великим і мережа може застрявати в локальних мінімумах, а при великих кроках можливо розбіжність процесу навчання або параліч мережі. Проблема ефективно вирішується адаптивним кроком, який для кожної ітерації дозволяє зробити крок, що мінімізує помилку мережі на даній ітерації. Існують методи, які на кожному тренувальному циклі (званому епохою) аналізують всю тренувальну вибірку і вибирають оптимальне значення і напрям кроку [12].

Одним з найбільш перспективних методів, що застосовуються на етапі навчання НМ, є генетичний алгоритм (ГА, по-англійськи - Genetic Algorithm, GA), що відноситься до еволюційних методів [6]. Генетичний алгоритм являє собою паралельний асинхронний оптимізаційний метод [6]. Пошук рішення в ньому здійснюється одночасно цілою популяцією

хромосом (хромосома - одиничне закодоване рішення задачі). Хромосоми можуть як обмінюватися досвідом один з одним, покращуючи свою придатність (оператор схрещування), так і освоювати нові області рішення (оператор мутації). Оптимізаційний критерій задає функцію оцінки придатності хромосом, і процес еволюції популяції являє собою поліпшення рішення задачі. Головною перевагою ГА є те, що при лінійному збільшенні розміру популяції, швидкість пошуку рішення зростає експоненціально (т.зв. прихований паралелізм) [6,12]. Це дозволяє досягти кращих субоптимальних рішень.

Оскільки навчання мережі - це оптимізаційний процес, генетичний алгоритм природним чином вкладається в алгоритм навчання НМ [6]. При цьому для прискорення ГА може бути введений оператор локального (градієнтного) спуску, що представляє собою метод зворотного поширення в застосуванні до окремої хромосоми-мережі. Так само, структура, закодована в хромосому мережі, і ввівши в оцінний критерій ГА штраф за надмірність архітектури, можна домогтися синтезу НМ з мінімально-необхідною архітектурою, яка має хороші узагальнюючі здібності [9].

Розглянемо застосування багатошарових нейронних мереж до розпізнавання людини по зображенню особи.

Найпростіше застосування одношарової НМ (автоасоціативною пам'яттю) полягає в навчанні мережі відновлювати зображення, що подаються. Подаючи на вхід невідоме зображення і обчислюючи якість реконструйованого зображення, можна оцінити, наскільки мережа розпізнала вхідне зображення. Позитивні властивості цього методу полягають в тому, що мережа може відновлювати перекручені і пошкоджені зображення, але для більш серйозних цілей він не підходить [7].

БНМ також використовується для безпосередньої класифікації зображень - на вхід подається або саме зображення в будь-якому вигляді, або набір раніше витягнутих ключових ознак зображення, на виході нейрон з максимальною активністю вказує приналежність до розпізнаного класу (рис.2.7). Якщо ця активність нижче деякого порога, то вважається, що поданий образ не належить ні до одного з відомих класів. Процес навчання встановлює відповідність поданих на вхід образів з належністю до певного класу. Це називається навчанням з учителем. В експериментах на тестовій базі ORL такий підхід дозволив досягти стабільної 93% точності розпізнавання (98% максимальної). У застосуванні до розпізнавання людини по зображенню особи такий підхід хороший для задач контролю доступу невеликої групи осіб. Він забезпечує безпосереднє порівняння мережею самих образів, але зі збільшенням числа класів час навчання і роботи мережі зростає експоненціально. Тому такі завдання, як пошук схожої людини у великій базі даних, вимагають вилучення компактного набору ключових характеристик, на основі яких можна здійснювати пошук.

Рис. 2.7 - Реконструкція за першими 28 з 10304 коефіцієнтів, зображення 92x112

НМ застосовується також для вилучення ключових характеристик зображень, що потім використовуються для подальшої класифікації. Показаний спосіб нейромережевої реалізації методу головних компонент. Суть методу головних компонент полягає в отриманні максимально декореллірованних коефіцієнтів, що характеризують вхідні образи. Такі коефіцієнти називаються головними компонентами і використовуються для статистичного стиснення і реконструкції зображень. При цьому невелике число коефіцієнтів використовується для представлення всього образу. Кожне зображення розкладається на лінійну комбінацію власних векторів. Для набору зображень облич власні вектори можуть бути представлені у вигляді зображень, такі зображення схожі на обличчя і називаються власними особами (eigenfaces, рис. 2.8).

НМ з одним прихованим шаром, що містить т нейронів, число яких багато менше, ніж розмірність зображення т << п, навчена за методом зворотного поширення помилки відновлювати на виході зображення, що подане на вхід, формує на виході коефіцієнти перших т головних компонент прихованих нейронів, які і використовуються для порівняння зображень. Архітектура такої мережі, званої рециркуляционною нейронною мережею (РНМ), показана на рис. 2.9.

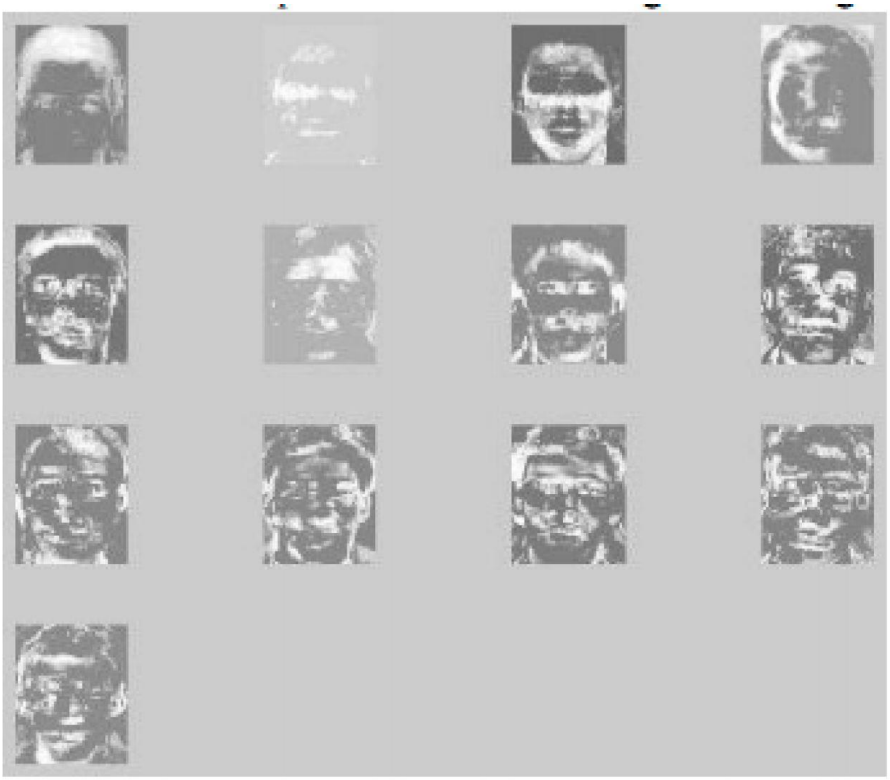


Рис. 2.8 - Власні особи (eigenfaces)

Зазвичай використовується від 10 до 200 головних компонент. Зі збільшенням номера компоненти її репрезентативність сильно знижується, і використовувати компоненти з великими номерами не має сенсу. Використання лінійних активаційних функцій в НМ дозволяє отримати на виході прихованого шару саме т перших головних компонент, аналогічних одержуваним при вирішенні матричних рівнянь. При використанні нелінійних активаційних функцій нейронних елементів можлива нелінійна декомпозиція на головні частини. Нелінійність дозволяє більш точно відобразити варіації вхідних даних, однак при цьому виходи прихованих нейронів будуть тільки схожі на головні компоненти. Ваги, що сформувалися при такому навчанні на вхідному і вихідному шарі, так само будуть схожі на власні особи, яким притаманне корисна властивість - існують компоненти, які в основному відображають такі суттєві характеристики особи, як стать, раса, емоції. Перші компоненти відбивають найважливіше - загальну форму обличчя, останні – різні дрібні відмінності між обличчями. Такий метод добре застосован для пошуку схожих зображень осіб у великих базах даних.

Цей метод також використовується в задачі виявлення особи на зображенні. Оцінюючи якість реконструкції вхідного зображення, можна дуже точно визначати його приналежність до класу осіб. Для зображень, які не є особами, реконструкція буде невисокої якості.

Переваги застосування РНМ для вилучення головних компонент перед рішенням матричних рівнянь:

- алгоритм навчання РНМ простий і універсальний;

- нелінійна активаційна функція дозволяє точніше реконструювати зображення;

- при вирішенні матричних рівнянь можливі проблеми, якщо приклади дуже схожі один на одного, РНМ позбавлена такого недоліку;

- не потрібно обчислювати всі власні вектори. Таким чином, час навчання мережі лінійно залежить від кількості видобутих головних компонент;

- для попередніх експериментів можна використовувати меншу кількість навчальних циклів, що знижує час навчання.

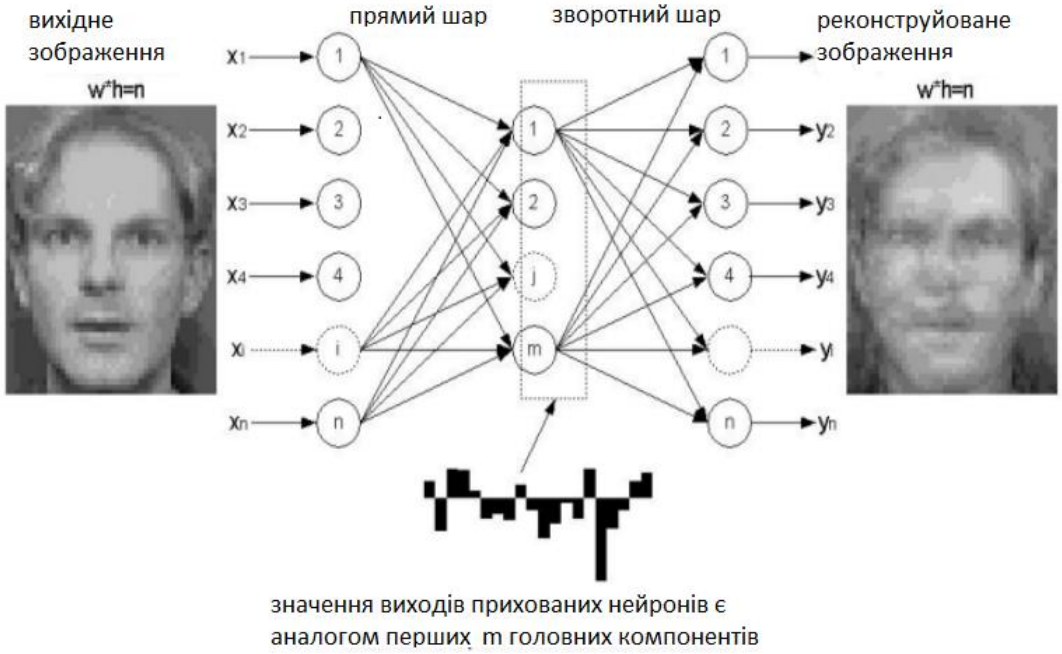


Рис. 2.9 - Архітектура нейронної мережі для вилучення головних компонент

Можливість подальшого зменшення розмірності головних компонент за допомогою НМ. Суть її полягає в використанні більшої кількості прихованих шарів, відповідальних за стиснення і реконструкцію зображення. Таке стиснення дозволяє вловити більш складні закономірності в наборі образів і, отже, представити їх точніше і меншим числом компонент. Зауважимо, що такий підхід може бути використаний і для вирішення деяких інших завдань, наприклад, для виділення (детекції) ділянок шкіри (областей обличчя та кистей рук людини на зображенні) для автоматизації розпізнавання жестової мови.

2.1.4 Нейронні мережі високого порядку і моментні НМ

Нейронні мережі високого порядку (НМВП, по-англійськи - High Order Neural Network) відрізняються від БНМ тим, що в них лише один шар, але на входи нейронів надходять так само терми високого порядку, що є добутком двох або більше компонент вхідного вектора, наприклад, для мереж другого порядку:

$$S = \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j - T \tag{2.2}$$

Такі мережі так само можуть формувати складні поверхні. Розділяють поверхню другого порядку (S = 0) називається гіперквадрікой [8]. Додаючи компоненти вхідного вектора в добуток, отримаємо клас поліноміальних поверхонь. Такі мережі також можна навчати за методом зворотного поширення. Багатошарові НМ в загальному випадку ефективніше, але існує ряд додатків, в яких мережі високого порядку краще ніж БНМ.

Перспективні архітектури та методи навчання нейронних мереж високого порядку і комбінованих нейронних мереж для розпізнавання зображень інваріантні до зсуву, масштабу і повороту. Архітектура таких мереж високого порядку заснована на обчисленні моментів зображення першими шарами.

Застосування НМВП третього порядку (рис. 2.10) для розпізнавання зображень облич, що мають довільні масштаб та орієнтацію зображення. Наведено методи навчання такої мережі. Особливості її полягають в тому, що для навчання певного класу досить пред'явити його образ без варіацій масштабів і поворотів - після навчання мережа буде розпізнавати відомі класи інваріантні до масштабу і поворотів зображення. Така мережа не є повно зв’язною, швидко навчається і працює. Відзначено істотне підвищення точності класифікації і масштабування повернених зображень такою мережею в порівнянні з БНМ.

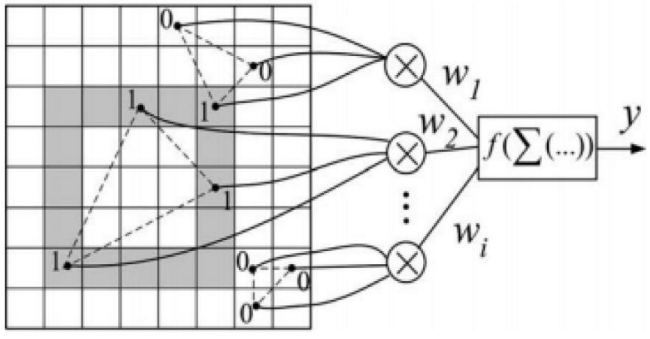


Рис. 2.10 - НМВП третього порядку

2.1.5 Радіально-базисні нейронні мережі

Радіально-базисні нейронні мережі (РБНМ, по-англійськи - Radial Basis Function Network, RBF) складаються з двох шарів (рис. 2.11).

Перший шар має радіально-базисну активаційну функцію:

$$y = \exp(\frac{-S^2}{2\sigma^2}) \tag{2.3}$$

де σ - середньоквадратичне відхилення, що характеризує ширину функції (розмір кластера), S визначається як відстань між вхідним і ваговим вектором:

$$S^2 = |X - W|^2 = \sum_i (x_i - w_i)^2 \tag{2.4}$$

являється відстанню до центру кластера, що визначаються конкретним нейроном. Таким чином, прихований шар являє собою набір кластерів в просторі образів і виконує перший етап кластеризації вхідного образу - значення активаційній функції кожного нейрона швидко зменшується з видаленням від центру кластера. Другий шар нейронів має лінійну активаційну функцію, і виконує другий етап кластеризації - розподіляє кластери по класах. На відміну від карток Кохонена тут обнулення нейронів, що не володіють максимальним вихідним значенням не потрібно, вони всі роблять внесок у класифікацію, і це перевага РБНМ.

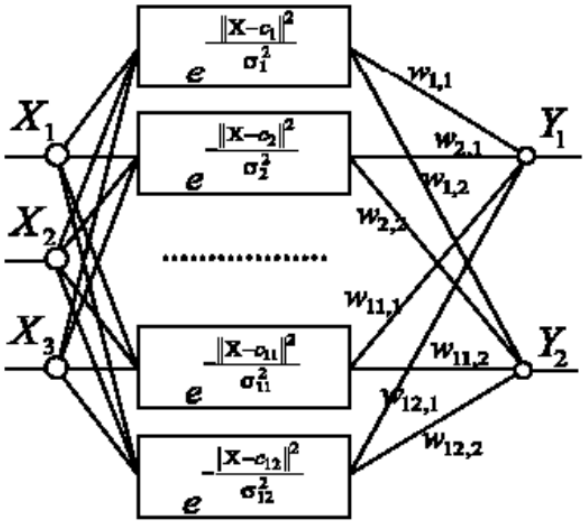


Рис. 2.11 - Радіально-базисна нейронна мережа,

Xi - елементи вхідного вектора,

Yi - елементи вихідного вектора, квадратами позначені радіально-базисні нейрони

РБНМ також здатна будувати складні області та апроксимувати багатовимірні функції. У порівнянні з багатошаровою нейронною мережею, радіально-базисна мережа навчається на порядок швидше, проте володіє набагато гіршою екстраполюючої здатністю, тобто нездатна працювати на образах, що лежать далеко від образів- прикладів. Розміри РБСН більше, ніж БНМ для аналогічних завдань, і РБНМ стають малоефективні з ростом розмірності вхідних даних.

Навчається така мережа в два етапи. Перший етап здійснюється без вчителя, на ньому перший шар виділяє компактно розташовані групи кластерів. При цьому коригуються центри кластерів. В даний час розроблені ефективні алгоритми, що дозволяють також підбирати оптимальний розмір кластерів для кожного нейрона [6] і отримувати оптимальну кількість нейронів в першому шарі [6]. На другому етапі навчання другий шар вчиться розподіляти вхідні образи, пропущені через перший шар, по класах. Інформація про еталонні значеннях виходів відома, навчання виконується з учителем. Таке навчання проводитися або матричними методами, або алгоритмом зворотного поширення помилки [12].

Способи вилучення ключових характеристик і дві різні архітектури РБНМ для розпізнавання осіб. У першому способі характеристики представляли собою набір головних компонент, у другому - коефіцієнти вейвлет перетворень. У першій архітектурі кількість виходів відповідала кількості класів, в другій застосовувався колектив мереж, кожна з яких була навчена розпізнавати тільки свій клас. Відзначено значні переваги класифікації РБНМ перед безпосереднім порівнянням ключових характеристик на тестовій базі MIT.

Різні архітектури ансамблів РБНМ для попередньої класифікації зображень. На вхід мережі надходило зображення цілком, на виходах формувалася проміжна класифікація, яка потім подавалася на вирішальні дерева для контекстно-орієнтованого розпізнавання зображень облич (наприклад: «знайти всі зображення певної людини, де вона в окулярах»). Різні мережі в ансамблях першої архітектури вчилися класифікувати зображення з різними типами змін, другої - з однаковими, але кількість нейронів змінювалася в процесі навчання. Вирішальний висновок робив «суддя» (нейронна мережа, навчена узагальнювати рішення колективу НМ), який приймав рішення на основі голосування ансамблю мереж.

Радіально-базисні мережі використовувалися для розпізнавання людини по набору геометричних характеристик і визначення його статі. Вихідними служили напівтонові і мальовані (карикатурні) зображення. Відзначено добру здатність РБНМ виділяти характерні ознаки.Використовувалася РБНМ спільно з оптичним потоком для аналізу емоційного виразу обличчя.

2.2 Топологічно впорядковане перетворення простору

Векторні квантователі і самоорганізуючі карти Кохонена (Self-Organizing Maps, SOM) використовуються для стиснення даних і вилучення ключових характеристик [12]. Так само вони служать основою для радіально-базисних мереж і когнітронів. Існують, однак, і підходи, які використовують карти Кохонена для вирішення оптимізаційних задач (наприклад, задачі комівояжера), що не виключає їх подальшого застосування для розпізнавання зображень.

Векторні квантователі вирішують задачу квантування і кластеризації даних [12]. Архітектура таких мереж складається з одного шару (не рахуючи вхідного розподільного), не має міжнейронних зв'язків, і є найпростішим варіантом карт Кохонена. Вхідний простір оптимальним чином розбивається на області-кластери. У процесі навчання відбувається виділення областей у вихідному просторі, кожному з яких відповідає окремий нейрон. Такі мережі функціонують за принципом «переможець бере все», активним вважається нейрон, що має найбільше вихідне значення:

$$S_j = \sum_i w_{ij} x_i = W_j^T X \tag{2.5}$$

тобто проекція вхідного вектора на ваги якого виявилася максимальною. Виходом мережі є номер нейрона-переможця.

Навчаються такі мережі без вчителя, тобто вони самі в процесі навчання вибирають оптимальне розбиття на області. Метод навчання таких мереж називається конкурентним. Після подачі навчального вектора вибирається нейрон, що має максимальну активність. Ваги такого нейрона змінюються в сторону відповідності вхідному вектору, наприклад:

$$W_j(t+1) = W_j(t) + \gamma(t)(X - W_j(t)) \tag{2.6}$$

Перед навчанням ваги ініціалізуються випадковими значеннями. Навчання ведеться до досягнення стабілізації ваг або завершується після певного числа ітерацій.

Для того щоб уникнути ситуації, коли деякі нейрони ніколи не можуть стати переможцями і для більш повного покриття простору (наприклад в областях де щільність образів вища, потрібна більша кількість нейронів) використовуються кілька підходів. По-перше, можна модифікувати ваги тих нейронів, хто програв з набагато меншою швидкістю. По-

друге, можна вести статистику перемог для кожного нейрона, і зменшувати можливості модифікації занадто частих переможців.

Для автоматизації визначення числа нейронів мережі, використовують такі алгоритми, як наприклад зростаючий нейронний газ. У ньому нейрони, що мають занадто велику кількість прикладів чи розмір осередку діляться на два нейрона [12].

Самоорганізуючі карти Кохонена дозволяють отримати топологічно впорядковане перетворення вихідного n- мірного простору в вихідний m-мірний, m << n. Архітектура такої мережі так само складається з одного шару, але нейрони в цьому шарі організовані в m-мірну решітку, і кожен нейрон має свою координату, що визначатиме її положення в решітці. Виходом мережі є координати нейрона-переможця.

На початок модифікації ваг

$$W_j(t+1) = W_j(t) + \gamma(t)h(t,i,j)(X - W_j(t)) \tag{2.7}$$

вводиться функція сусідства h, спадаюча з відстанню між нейронами:

$$h(t,i,j) = \exp\left(\frac{-|i-j|^2}{2\sigma^2(t)}\right) \tag{2.8}$$

де i - нейрон-переможець, j – нейрон, що модифікується, j – i - відстань між ними, σ(t) - радіус області сусідства, убуває з часом в процесі навчання. Для прискорення навчання використовуються зростаючі мережі, коли заново створювані області решітки ініціалізуються значеннями сусідніх нейронів.

Таким чином, навчена мережа здатна топологічно впорядковано відображати вхідний простір в вихідний - вектори, близькі в вихідному простору, матимуть близькі координати нейронів в решітці.

Це є особливо корисним при класифікації даних, що мають велику кількість класів. Наприклад, при класифікації локальних ділянок зображень, може бути дуже велике число класів, в яких перехід від одного класу до іншого практично безперервний, ускладнюючи визначення меж класів.

Тривимірна карта Кохонена (по п'ять вузлів на кожний вимір) застосовувалася для зменшення розмірності локальних ділянок 5x5 (розмірність 25) зображень осіб (рис. 2.12). Кожній ділянці зображення 5x5 відповідає своя координата в карті Кохонена.

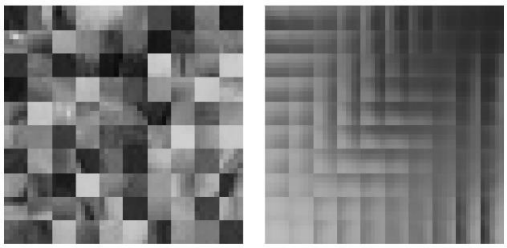


Рис. 2.12 - Застосування карт Кохонена для зменшення розмірності ділянок зображень облич. Зліва - топографічна карта ділянок зображень до, праворуч - після навчання.

Після навчання схожі ділянки мають близьке положення на мапі. Вхідне зображення відображається на один зі 125 вузлів, положення якого в тривимірній решітці кодує вектор вихідного простору. Три виміри карти використовуються в якості осей трьох ключових характеристик (features). Таке перетворення забезпечує часткову стійкість до зміни освітлення, зсувам та спотворень, позбавляє від необхідності попередньої обробки зображення (перевага - прискорення роботи), а так само значно прискорює процес навчання і класифікації, роблячи цю систему придатною в системах, що працюють в реальному масштабі часу. У цій роботі значення виходів карт Кохонена використовувалися для подальшого розпізнавання по зображеннях осіб. Відзначено невелику перевагу карт Кохонена перед методом головних компонент.

Карти Кохонена також застосовувалися для виявлення очей на зображенні особи. Карта навчалася на типових прикладах зображень очей. Наявність очей в уже згадуваній ділянці зображення визначалося по карті активності всіх нейронів, в цьому випадку на ній спостерігалися характерні піки, як показано на рис.2.13:

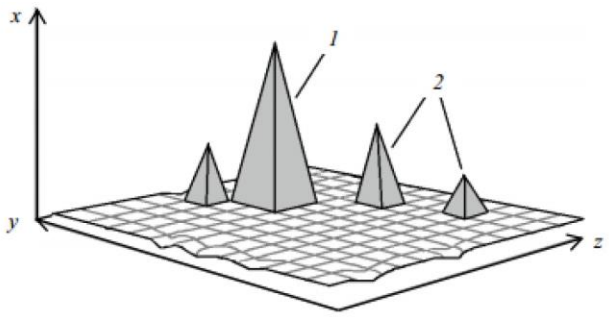


Рис. 2.13 Відгуки нейронної мережі. 1 - найбільш вірогідне положення очей, 2 - менш ймовірні місця розташування (шуми)

2.3 Розпізнавання з урахуванням топології простору

Описувані такі типи нейронних мереж дозволяють врахувати топологію простору зображення. Адже зображення - це не просто н- мірний вектор, складений з яскравості пікселів. Зображення має свою топологію, яка визначається через двовимірне локальне сусідство пікселів [12].

Принципи роботи таких мереж ґрунтуються на необхідності розділення зображення на маленькі ділянки і ієрархічному зіставленні як взаємного їх розташування, так і змісту. Такі мережі є найбільш перспективними для розпізнавання зображень.

2.3.1 Когнітрон

Когнітрон був розроблений ґрунтуючись на анатомії і фізіології мозку, і своєю архітектурою схожий на будову зорової кори. Кожен шар мозку реалізує різні рівні узагальнення: вхідний шар чутливий до простих образів, таких, як лінії, і їх орієнтації в певних областях візуальної області, в той час як реакція інших верств є більш складною, абстрактною і незалежною від позиції образу. Аналогічні функції реалізовані в когнітроні шляхом моделювання організації зорової кори [12].

Головні архітектурні відмінності когнітрону полягають в тому (рис. 2.14), що кожний нейрон пов'язаний лише з невеликою локальною областю попереднього шару, і такі області перекриваються одна з одною. Шарів в когнітроні зазвичай більше ніж в мережах інших типів. Таким чином досягається ієрархічна організація, коли на вищих шарах когнітрон реагує на більш абстрактні образи, менше реагує на їх усунення і спотворення. Навчається когнітрон конкурентним навчанням (без вчителя).

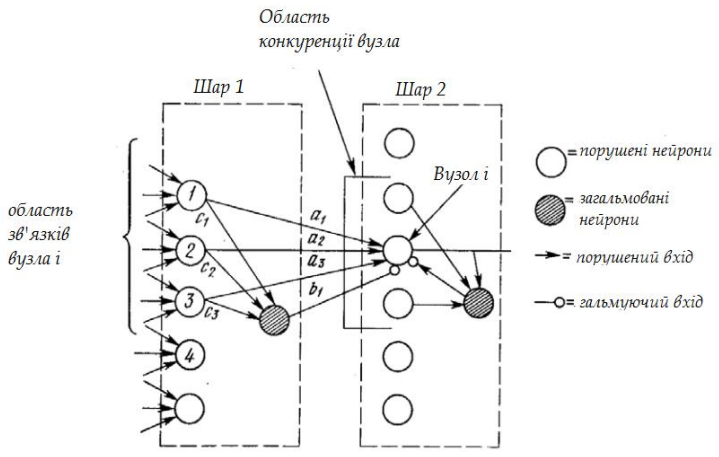


Рис. 2.14 – Когнітрон

2.3.2 Неокогнітрон

У зоровій корі були виявлені вузли, що реагують на такі елементи, як лінії і кути певної орієнтації. На більш високих рівнях вузли реагують на більш складні і абстрактні образи такі, як кола, трикутники і прямокутники. На ще вищих рівнях ступінь абстракції зростає до тих пір, поки не визначається вузли, що реагують на обличчя і складні форми. У загальному випадку вузли на більш високих рівнях отримують вхід від групи низькорівневих вузлів і, отже, реагують на більш широку область візуального поля. Реакції вузлів більш високого рівня менш залежать від позиції і більш стійкі до спотворень.

Неокогнітрон є подальшим розвитком ідеї когнітрону і більш точно відображає будову зорової системи, дозволяє розпізнавати образи незалежно від їх перетворень: зміщення, обертання, зміни масштабу і спотворення. Неокогнітрон може як самонавчатися, так і навчатися з учителем. Неокогнітрон отримує на вході двовимірні образи, аналогічні зображенням на сітчастій оболонці ока, і обробляє їх в наступних шарах аналогічно тому, як це було виявлено в зоровій корі людини.

Головна відмінність неокогнітрона від когнітрону - це двовимірна організація локальних ділянок і площинна ієрархічна структура (рис. 2.15).

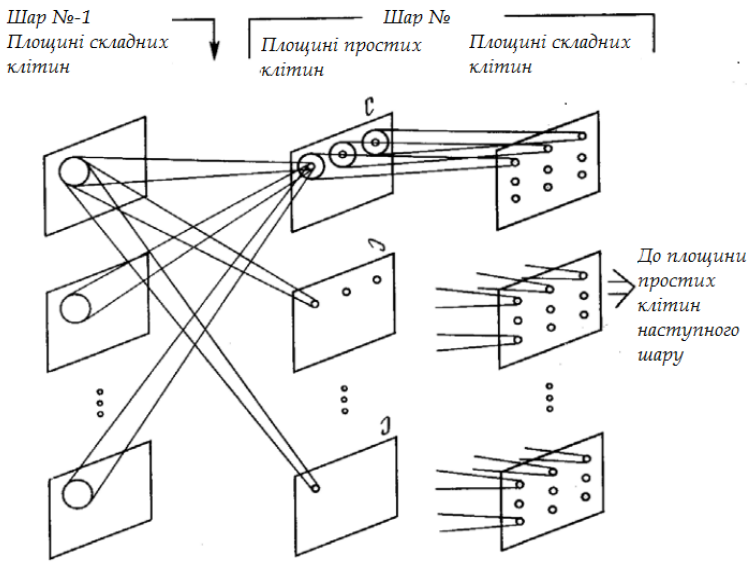


Рис. 2.15 – Неокогнітрон

Кожен шар складається з площини простих і складних клітин. Кожен нейрон простий площині пов'язаний з локальною двовимірною ділянкою площини попереднього шару, ваги всіх нейронів в межах однієї площини однакові, і таким чином площина реагує на певний спосіб, що

знаходиться в ділянці зображення (приклад на малюнку - площині реагують на букву «С» повернену під різними кутами). Становище активованого таким чином нейрона в простій площині зазначає ділянку, в якому знайдений цей образ, незалежно від спотворення цього образу. Нейрон комплексній площині пов'язаний з ділянкою своєї простий площині, і виявляє активність нейронів на цій ділянці, зменшуючи таким чином чутливість до позиції образу.

Таким чином досягається ієрархічна обробка зображення, коли на наступних шарах неокогнітрон реагує на більш загальні риси зображення не збиваючись на спотворення, зсув і т.д. Класичний неокогнітрон є потужним засобом розпізнавання зображень, однак вимагає високих обчислювальних витрат, які на сьогоднішній день недосяжні.

Однак існує безліч робіт, наприклад, спрямованих на удосконалення неокогнітрона. Один з найбільш перспективних підходів для розпізнавання людини по зображенню особи - це згорточні нейронні мережі [13].

2.3.3 Згорточні нейронні мережі

У класичній багатошаровій нейронній мережі міжшарові нейронні з'єднання повнозв'язні, і зображення представлено у вигляді і-мірного вектора, який не враховує ні двовимірну локальну організацію пікселів, ні можливостей деформації. Архітектура згорточної НМ (рис. 2.16) спрямована на подолання цих недоліків і ґрунтується на принципах архітектури неокогнітрона, спрощеного і доповненого навчанням алгоритмом зворотного поширення помилки [12].

У ній використовувалися локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів), спільні ваги (забезпечують детектування деяких рис в будь-якому місці зображення) і ієрархічна організація з просторовими підвибірками (spatial subsampling).

Згорточна НМ (ЗНМ, Convolutional Neural Network) забезпечує часткову опірність змін масштабу, зсувів, поворотів, зміни ракурсу і іншим спотворень.

Архітектура ЗНМ (рис. 2.16), складається з багатьох верств. Шари бувають двох типів: згорточні (Convolutional) і подвибіркові (Subsampling), згорточні і подвибіркові шари чергуються один з одним. У кожному шарі є набір з декількох площин, причому нейрони однієї площини мають однакові ваги, провідні до всіх локальних ділянках попереднього шару (як в зоровій корі людини), зображення попереднього шару як би сканується невеликим вікном і пропускається крізь набір ваг, а результат відображається на відповідний нейрон поточного шару.

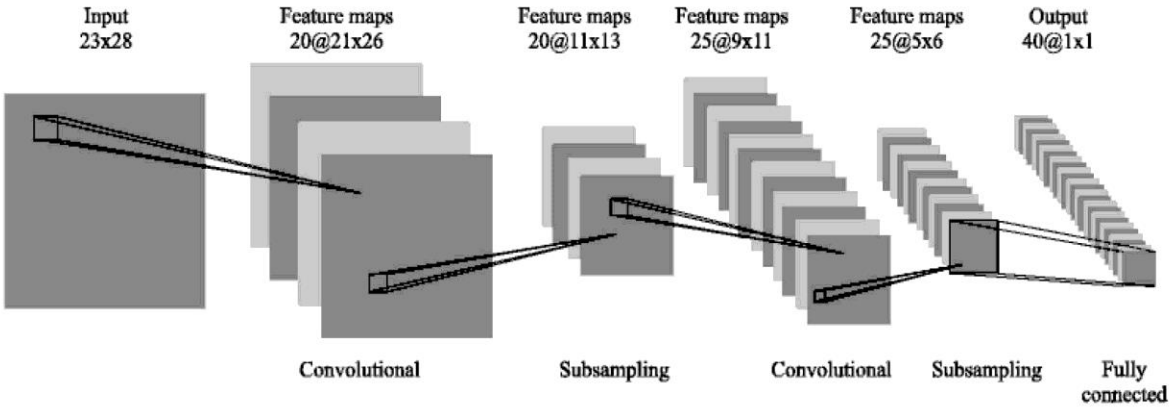


Рис. 2.16 - Архітектура згорточної нейронної мережі

Таким чином, набір площин являє собою карти характеристик (feature maps) і кожна площина знаходить «свої» ділянки зображення в будь-якому місці попереднього шару. Наступний за згорточним шаром подвибірковий шар зменшує масштаб площин шляхом локального усереднення значень виходів нейронів. Таким чином, досягається ієрархічна організація. Наступні шари витягують більш загальні характеристики, менше залежать від спотворень зображення. Навчається ЗНМ стандартним методом зворотного поширення помилок. Порівняння БНМ і ЗНМ показало істотні переваги останньої як за швидкістю, так і по надійності класифікації. Корисною властивістю ЗНР є і те, що характеристики, які формуються на виходах верхніх шарів ієрархії, можуть бути застосовні для класифікації за методом найближчого сусіда (наприклад, обчислюючи Евклідову відстань), причому ЗНМ може успішно використовувати такі характеристики і для образів, відсутніх в навчальному наборі. Для ЗНМ характерні висока швидкість навчання і роботи. Тестування ЗНМ на базі даних ORL, що містить зображення осіб з невеликими змінами освітлення, масштабу, просторових поворотів, положення і різними емоціями, показало 98% -ву точність розпізнавання.

2.4 Аналіз характеристик нейронних мереж для розпізнавання зображень

Для мобільних додатків розмір 100 мегабайт є дуже великим. При цьому розмір більшості нейронних мереж (добре спроектованих і показуючих високу точність результатів), може досягати понад п'ятсот мегабайт. Якщо брати до уваги те, що більшість магазинів для додатків встановлює обмеження на розмір публікованих додатків в 100 мегабайт, то опублікувати додаток в яких використовується нейронна мережа розміром 500 мегабайт є дуже складним завданням. Все вищесказане робить неможливим використання таких об'ємних нейронних мереж на мобільних платформах. Нижче на малюнку представлений графік найпопулярніших нейронних мереж, що відображає їх розмір (рис.2.17).

Як видно з графіка, GoogleNet єдина нейронна мережа, яка вписується в рамки сучасних магазинів додатків. Саме нею ми і користуємося в розробці поточного проекту, але це не означає що не потрібно вживати ніяких заходів по її оптимізації, так як швидкість роботи все ще залишається важливою частиною всієї системи. І розмір 80 мегабайт все ще залишається досить великим, адже крім нейронної мережі в додатку буде присутній безпосередньо код програми, графічні дані та інші додаткові ресурси. Виходить, що при розмірі нейронної мережі в 80 мегабайт, потрібно дуже постаратися спроектувати додаток, який в загальній сумі буде займати місця менше ніж 100 мегабайт. При цьому доведеться оптимізувати код програми і жертвувати якістю графічних ресурсів.

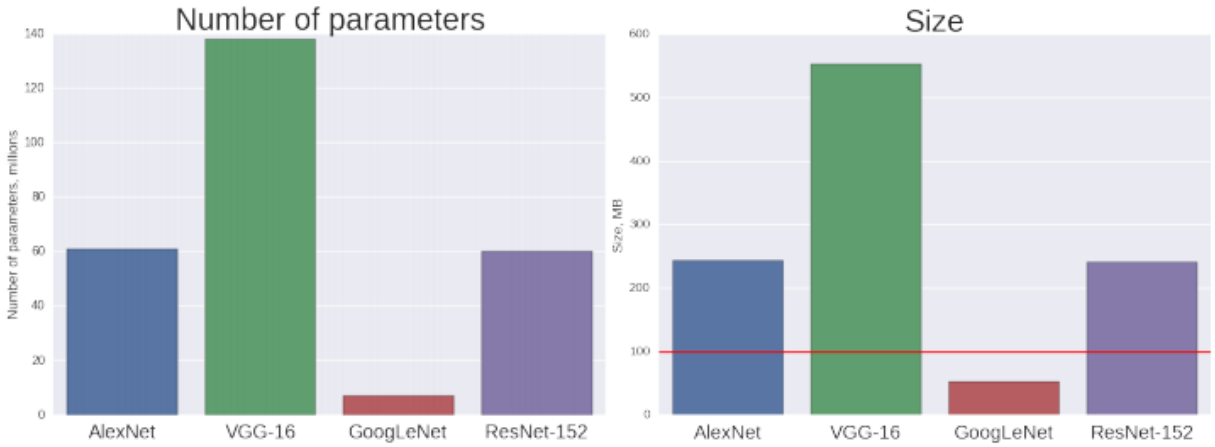


Рис 2.17 – Характеристики нейронних мереж

Для вирішення цієї проблеми існує кілька підходів. В рамках цієї роботи буде розглянуто два підходи по оптимізації нейронних мереж, які будуть представлені нижче.

2.4.1 Квантизація

Під час тренування нейронної мережі, у більшості випадків, використовуються параметри типу float, для того щоб всі маленькі зміни градієнта правильно додавалися або віднімалися від ваг. Однак, після правильного навчання мережа починає володіти такою властивістю як стійкість. Тобто, незначна зміна на вході проявляється незначними змінами на результаті. Відповідно і в проміжних шарах незначні зміни будуть незначно впливати на результат. Це означає, що ми можемо так перетворити ваги, і так перетворити дані, які проходять по мережі, що вони будуть дискретними. Таким чином, ми можемо замість використання змінних float, розбивши рівномірно сіткою, використовувати змінні типу int, або навіть, більш того, short. У використанні цього методу немає ніяких складнощів. Можна знайти

велику кількість джерел вже з реалізованим алгоритмом і просто скористатися найбільш підходящим. Квантування відкриває великі можливості для використання нейронних мереж в системах, в яких взагалі немає змінних з плаваючою комою. Це вирішує проблему запуску нейронної мережі в системах, в яких використовуються тільки цілочисельні обчислення.

На рисунку 2.18 зображено результат квантизації.

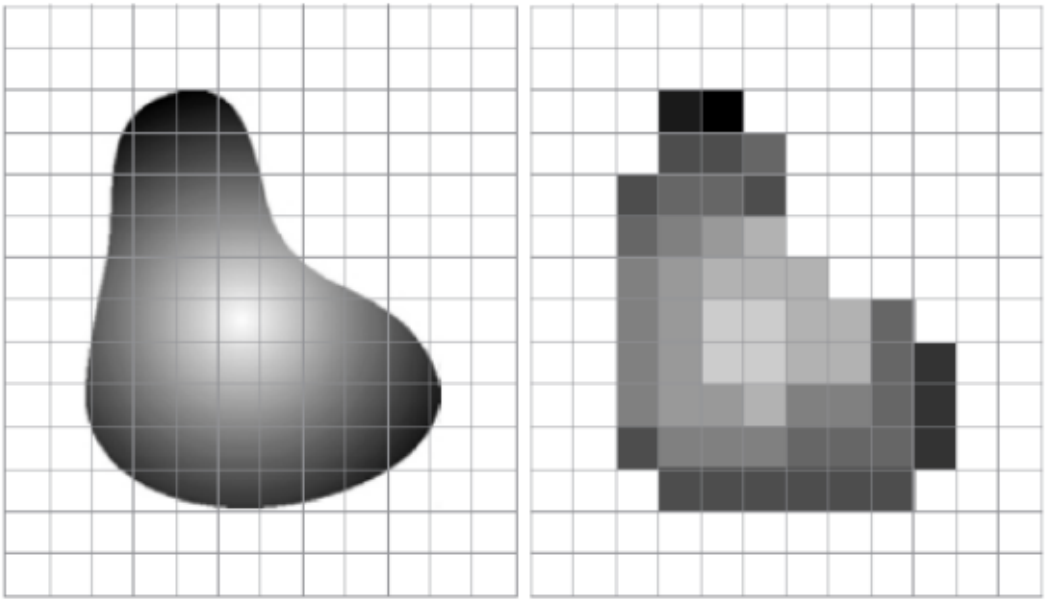


Рис. 2.18 – Приклад квантизації

2.4.2 Спрощення (pruning)

Ідеєю даного методу є те, що зв'язки в нейронних мережах виявляються надлишковими і представляється можливим прибрати частину з цих зв'язків без втрати якості. Складність полягає у визначенні, який конкретно зв'язок можна прибрати, а який необхідно залишити. Існує ряд підходів для визначення цього. Найпростіший і найбільш часто вживаний це видалення ваг, які дорівнюють нулю, так як ці значення ніяк не вплинуть на результат. На графіку нижче представлено розподіл ваг для нейронної мережі VGG-16.

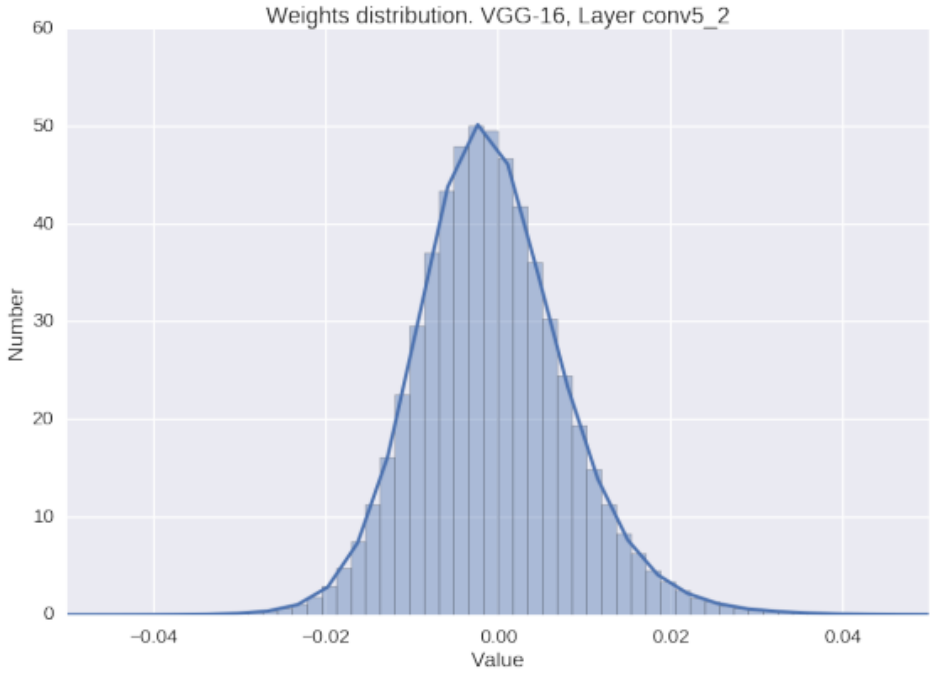


Рис. 2.19 – Розподіл ваг мережі VGG-16

Перше, що приходить на розум при погляді на представлений графік це вирізати всі ваги за абсолютним значенням, яке дорівнює нулю. Такий підхід є базовим і інтуїтивним. Однак ще в 1990 році Ян Лі Кун опублікував статтю "Optimal brain damage", в якій довів, що такий підхід не завжди видаляє потрібні ваги і потрібно вираховувати матрицю похідних за вагами і видаляти ту, в якій друга похідна дорівнює нулю. Такий підхід є досить складним для реалізації і вимагає значних обчислювальних витрат. Було розроблено безліч методів для оптимізації роботи цього підходу, але з 2015 року стали застосовувати підхід, при якому вирізаються всі ваги рівні від -0.01 до 0.01, перетворивши Dance матрицю в Sparse матрицю. Другим етапом спрощення нейронної мережі є тонка настройка ваг (fine-tuning). Після тонкої настройки ваг необхідно провести тестування нейронної мережі і, якщо її точність недостатня, то необхідно повторити кілька циклів. Вся схема спрощення нейронної мережі зображена на рисунку 2.20.

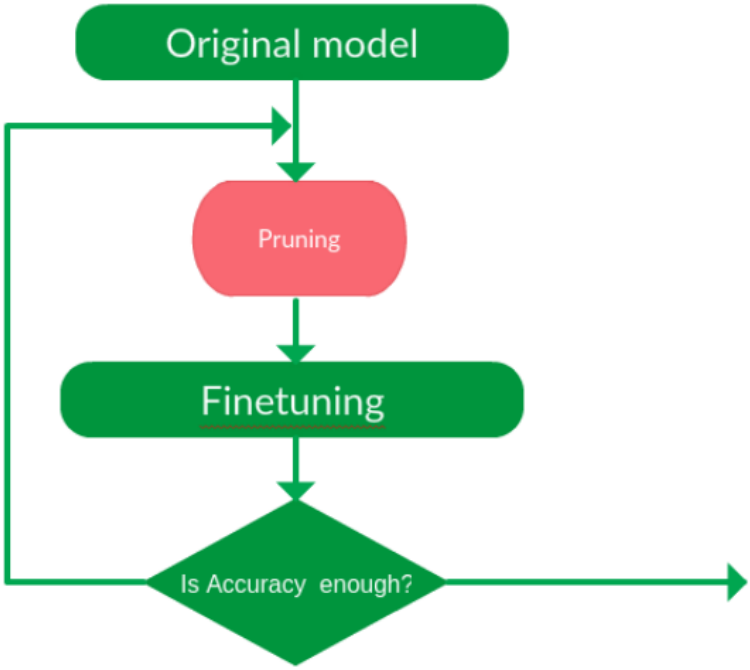


Рис. 2.20 – Алгоритм спрощення

Саме цим методом користувалися при оптимізації нейронної мережі в даній роботі. Для того щоб продемонструвати його якості, нижче наведено кілька прикладів застосування спрощення нейронної мережі для деяких нейронних мереж. На рисунку 2.21 буде проілюстровано графік застосування методу спрощення нейронної мережі на мережі VGG-16.

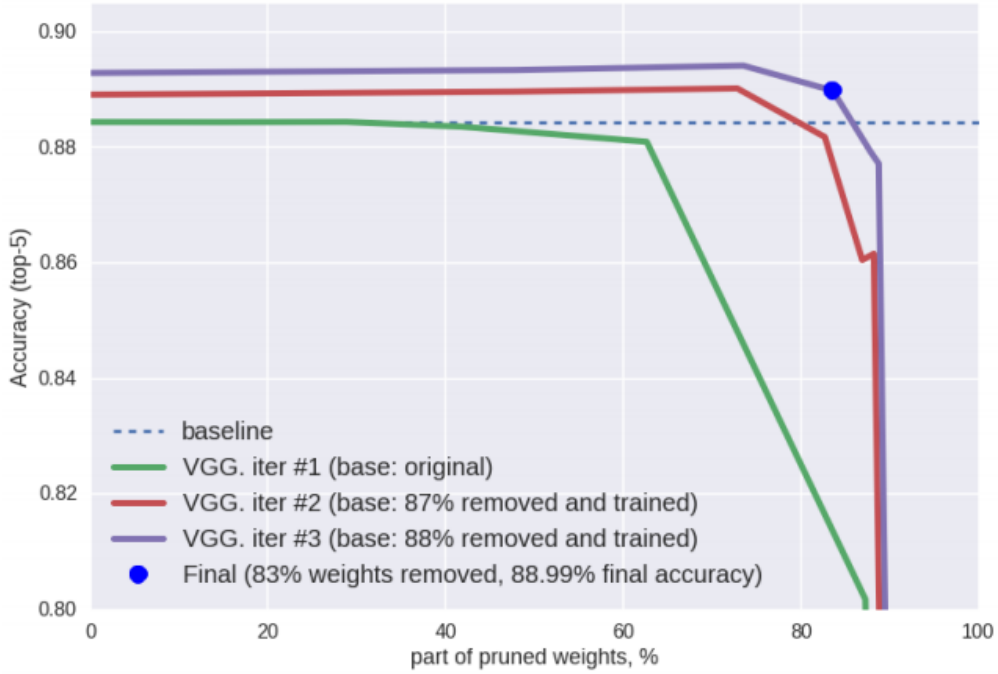


Рис. 2.21 – Спрощення мережі VGG-16

На осі Y знаходиться коефіцієнт точності нейронної мережі, а по осі X кількість вирізаних даних у відсотках. Пунктиром позначено базову точність нейронної мережі, тобто її точність без спрощення. Як видно на графіку, після проведення декількох циклів спрощення проявляється неймовірна властивість, спрощена модель демонструє більш високу точність у порівнянні з повною, і при цьому має розмір на 88% менше, ніж до спрощення. Це дуже хороший показник, але з огляду на великий початковий розмір мережі в 500 мегабайт, то після спрощення отримали мережу розміром 110 мегабайт, що все одно є занадто великим розміром для застосування на мобільному пристрої.

Нижче представлений графік розподілу ваг після спрощення нейронної мережі VGG-16 (рис.2.22).

За поданим графіком видно, що нейронів зі значенням 0 не залишилося. Так само видно, що розподіл втратил симетрію і зрушила трохи вліво. Цей ефект вийшов після проведення тонкої настройки ваг. Ще одним цікавим аспектом проведення спрощення нейронної мережі є така її властивість, що якщо продовжити вирізати значення близьких до нуля і вводити більш жорсткі рамки для вирізання ваг, то отримаємо мережу в якій всі ваги будуть рівні 1 або -1. Це дозволяє уникнути застосування операції множення, тобто всі операції в пакунку замінюються на додавання чи віднімання. Відповідно, отримуємо величезне прискорення.

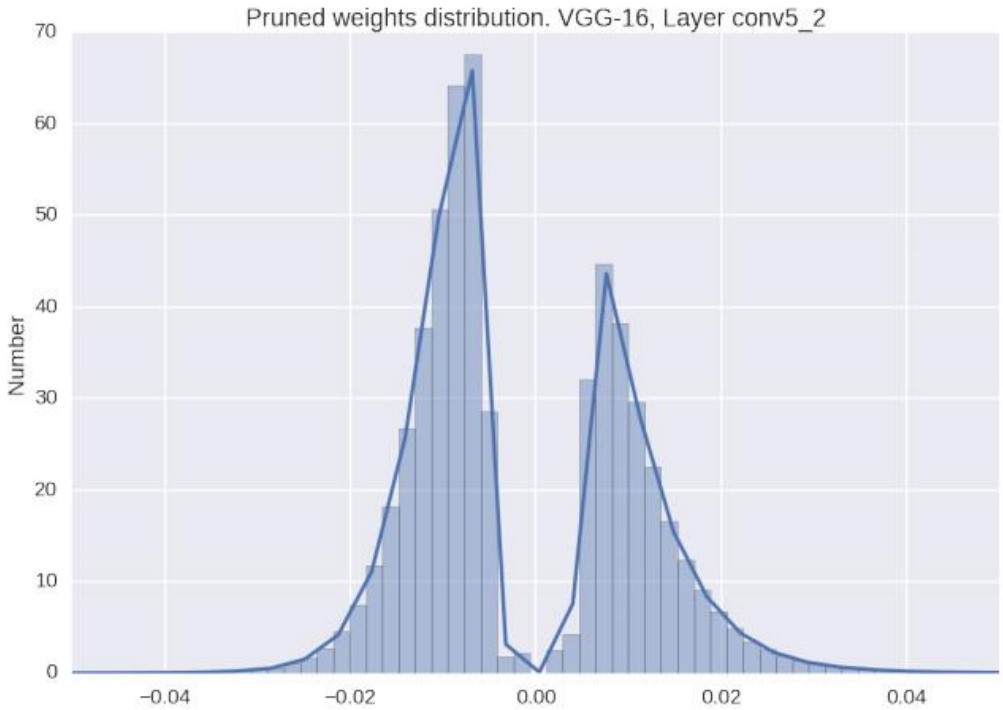


Рис. 2.22 – Графік розподілу ваг після спрощення

Далі представлено графік спрощення нейронної мережі ResNet (рис.2.23).

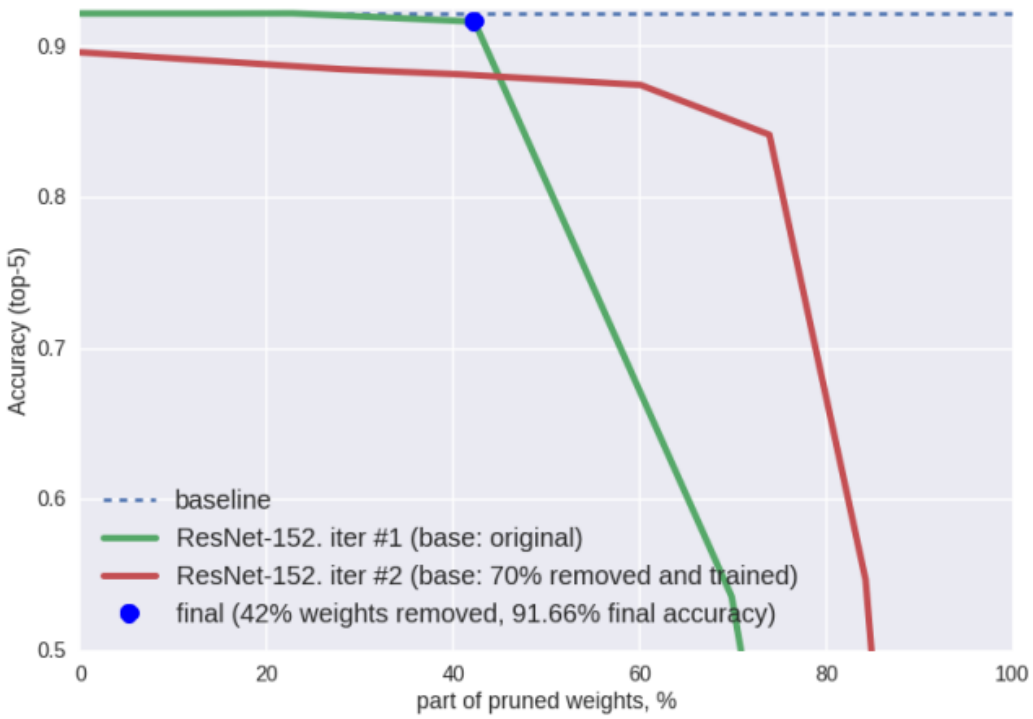


Рис. 2.23 – Спрощення мережі VGG-16

Як видно з графіка спрощення даної нейронної мережі не призводить до отримання таких же неймовірних результатів. Більш того, тренування цієї нейронної мережі займає величезну кількість часу. Спрощення стає дуже складним і трудомістким. За допомогою спрощення вдалося скоротити розмір нейронної мережі на 40%, що теж досить високий показник.

2.5 Висновки до розділу

У цьому розділі були досліджені основні методи аналізу зображень та методи оптимізації нейромереж, детально розглянуті аналоги найпопулярніших нейромереж. Для реалізації програмного продукту було обрано метод з використанням згорточної мережі. Тому що такі нейронні мережі працюють швидко займають менше місця і демонструють високу точність розпізнавання, а це саме те що необхідно для роботи на мобільному пристрої.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

У розділі розроблено програмний продукт - додаток під операційну систему Android. Додаток використовує відеопотік з камери пристрою в якості вхідного потоку даних. З цього вхідного потоку безперервно вибирається зображення, аналізуються і після виведення результату аналізу ітерація повторюється. Таким чином, частота вибору зображення залежить від швидкості роботи алгоритму. Як результат, виводиться назва класу і коефіцієнт в проміжку від нуля до одиниці, що характеризує ймовірність приналежності об'єкта на зображенні до вищезазначеного класу.

Для реалізації програми використовувалася середовище розробки Android Studio, бібліотека tensorflow і збирач проектів Bazel. Так як tensorflow немає підтримки мови Java, то довелося використовувати Android ndk і всі методи взаємодії з tensorflow реалізовані на мові програмування C ++. Але при цьому всі інтерфейси і частини не пов'язані з tensorflow написані на Java, так як Java є стандартною мовою для розробки Android додатків.

В якості моделі нейронної мережі була використана InceptionV3, заснована на GoogleNet. Дана модель перемогла на змаганні ImageNet (щорічне змагання з комп'ютерного зору) в 2012 році.

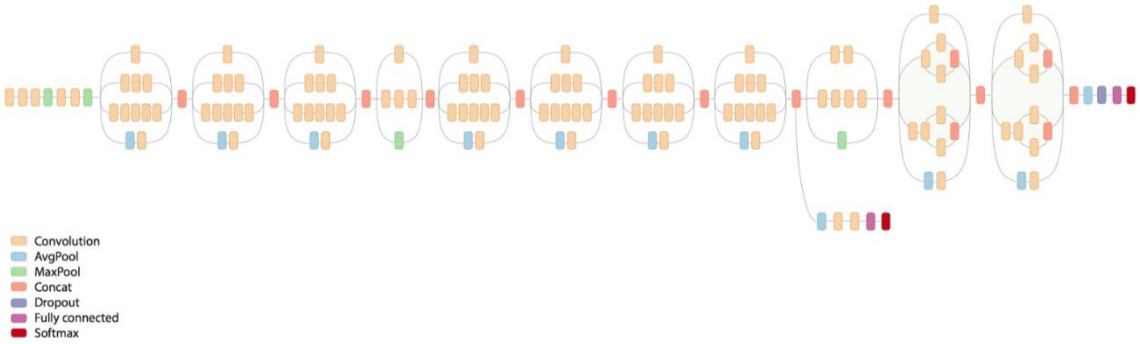


Рис 3.1 – Структура Inception-v3 моделі

Ця нейронна мережа здатна класифікувати понад 1000 об'єктів, а її коефіцієнт помилок становить 3,68%.

Для тренування мережі був використаний готовий алгоритм, реалізований мовою Python. Для прикладу мережа була натренована на розпізнавання деяких видів квітів і звичайної чашки. Тренувальний dataset був підготовлений з зображень квітів взятих з мережі інтернет (троянди і соняшники) і зображень різних чашок, зроблених безпосередньо на камеру. В результаті було

зібрано близько 300 зображень для кожного класу. Зображення чашок робилися для чашок різної форми, кольору, на різному фоні і при різному освітленні. Це допомогло поліпшити якість роботи нейронної мережі.

У таблиці 3.1 зображені данні відповідності классів до їх найменувань у нейронній мережі і до виду та кількості зібраних тренувальних матеріалів.

Таблиця 3.1 – Класи об’єктів у системі

Клас	Ім’я класу у системі	Вид сбору тренувальних данних	Кількість тренувальних одиниць
Троянда	roses	У мережі інтернет	358
Соняшник	sunflowers	У мережі інтернет	406
Чашка	dntl_cup	Аматорська фотозйомка	312

Як видно з таблиці, при зборі даних використовувалося два методу:

1) Звичайний пошук зображень в інтернеті.

2) Зйомка зображень безпосередньо на камеру мобільного пристрою або фотоапарата. Також можна побачити, що кількість відібраних даних приблизно однакова. Теоретично, способи збору інформації не повинні сильно вплинути на якість результатів роботи програми, але в мережі інтернет можна знайти більш різноманітні знімки, зроблені на різні фотокамери професійні або аматорські, зроблені в різних місцях з різним освітленням і так далі. При використанні ж другого методу спектр зображень істотно знижується, так як використовується не дуже велика різноманітність видів чашок, використовується при цьому всього лише два різновиди камер, більш того кількість локацій де можна зробити фотозйомку обмежена. З іншого ж боку, при самостійній зйомці об'єкта можна знімати кожен об'єкт з усіх можливих ракурсів, в той час як в інтернеті, здебільшого, всі фотографії однотипні, складно підібрати таку кількість, щоб покрити всі можливі варіанти положення об'єкта в просторі. Далі ще буде проведено аналіз та порівняння результатів на виявлення впливу вибору типу відбору тренувальних даних на якість роботи програми.

3.1 Підготовка програмного забезпечення

Для роботи всіх вищезазначених модулів необхідно виконати наступні кроки:

- Клонувати tensorflow проект з (https://github.com/tensorflow/tensorflow).

- Знайти coded_stream.h в / tensorflow / Google / Protobuf / SRC / Google / Protobuf / Io розділ $ HOME вашого tensorflow будувати і змінити значення з 64 до 256.

- Bazel, Android NDK і Android SDK все повинні бути встановлені d системі.

- Отримати рекомендовану версію Базел знайти за адресою:

- В <workspace_root> / Workspace повинні бути розкоментовані та заповнені відповідним чином залежно від того, де встановили NDK і SDK. В іншому випадку помилки, такі як: "The external label '//external:android/sdk' is not bound to anything" буде відображене.

- Якщо виводяться помилки побудови про буфери протоколу, потрібно запустити команду git submodule update --init і побудувати заново.

3.2 Тренування tensorflow - мережі для розпізнавання об'єктів на зображенні

Для тренування мережі потрібно:

- проект tensorflow;

- Bazel;

- набір зображень для тренування.

Велика частина інформації є в туторіали від tensorflow. У ньому описана процедура тренування мережі на готовому наборі зображень (квіти).

В першу чергу потрібно зробити білд проекту тренування командою: «bazel build tensorflow / examples / image_retraining: retrain»

Далі необхідно підготувати колекцію зображень. Створити папку де будуть знаходиться зображення. Всередині цієї папки для кожного об'єкта створити директорію з ім'ям цього об'єкта. Відповідно, помістити зображення в підготовлені папки. При створенні колекцій зображень слід пам'ятати кілька простих правил:

- для кожного об'єкта, який потрібно розпізнати необхідно не менше 100 різних фото. Чим більше фотографій, тим вище якість розпізнавання;

- фото повинні бути зроблені в різних контекстах (в різних приміщеннях, на вулиці, з різним освітленням, на різному фоні). Все це необхідно для того, щоб алгоритм міг відокремити контекст, який можна проігнорувати, від самого об'єкта;

- не можна поміщати в одну категорію об'єкти, які докорінно відрізняються один від одного. Іншими словами потрібно уникати узагальнення, наприклад не варто створювати

категорію транспорт, доцільніше створити категорію для кожного конкретного транспортного засобу (автомобіль, мотоцикл ...);

- не можна використовувати зображення на яких розміщені об'єкти з різних категорій.

Далі необхідно запустити тренувальний скрипт:

«bazel-bin / tensorflow / examples / image_retraining / retrain --image_dir / ~ / your_objects_photos»

Скрипт згенерує файл натренованої моделі (/tmp/output_graph.pb) і файл зі списком імен об'єктів (/tmp/output_labels.txt).

Для підвищення якості тренування мережі можна вказати додаткові аргументи до команди:

- how_many_training_steps: кількість тренувальних кроків. За замовчуванням 4000;

- random_crop, --random_scale, --random_brightness: випадкове додавання ефекту до фотографії. Вказується у відсотках (1-100);

- flip_left_right: випадковим чином розділяє зображення на дві частини і додає дзеркальне відображення.

3.3 Оптимізація нейромережі

Дуже важливими характеристиками для нейронних мереж є швидкість їх роботи і обчислювальна продуктивність. З огляду на те, що спроектована нейронна мережа буде працювати на мобільному пристрої, важливість цих характеристик виростає удвічі. Більш того, ще однією важливою характеристикою в даній ситуації є розмір нейронної мережі.

Для вирішення цієї проблеми доведеться відокремити нейронну мережу від самого додатка і після установки користувач буде завантажувати мережу окремо. Відповідно, на роботу такої нейронної мережі буде потрібно витратити значно більшу кількість ресурсів. Вузьким місцем при проектуванні мобільних систем є обмін даними між оперативною пам'яттю і кешем, чим менше даних тим система буде швидше працювати.

Для розробки даної роботи була застосована нейронна мережа GoogleNet. Нижче представлений графік спрощення нейронні мережі.

Як видно з графіка (рис. 3.2), вдалося домогтися досить високих результатів. Нехай якість нейронної мережі знизилася, але вона знизилася незначно, мова йде, всього лише, про тисячних частках відсотка. При цьому розмір нейронної мережі вдалося знизити на 58%. Таким чином, вихідну нейронну мережу розміром 85 мегабайт вдалося скоротити до 36 мегабайт. Це дуже високий показник. Загальний розмір АПК (APK(англ. Android Package) — формат

архівних файлів-додатків для «Android».) файлу програми разом з нейронною мережею становить 50 мегабайт.

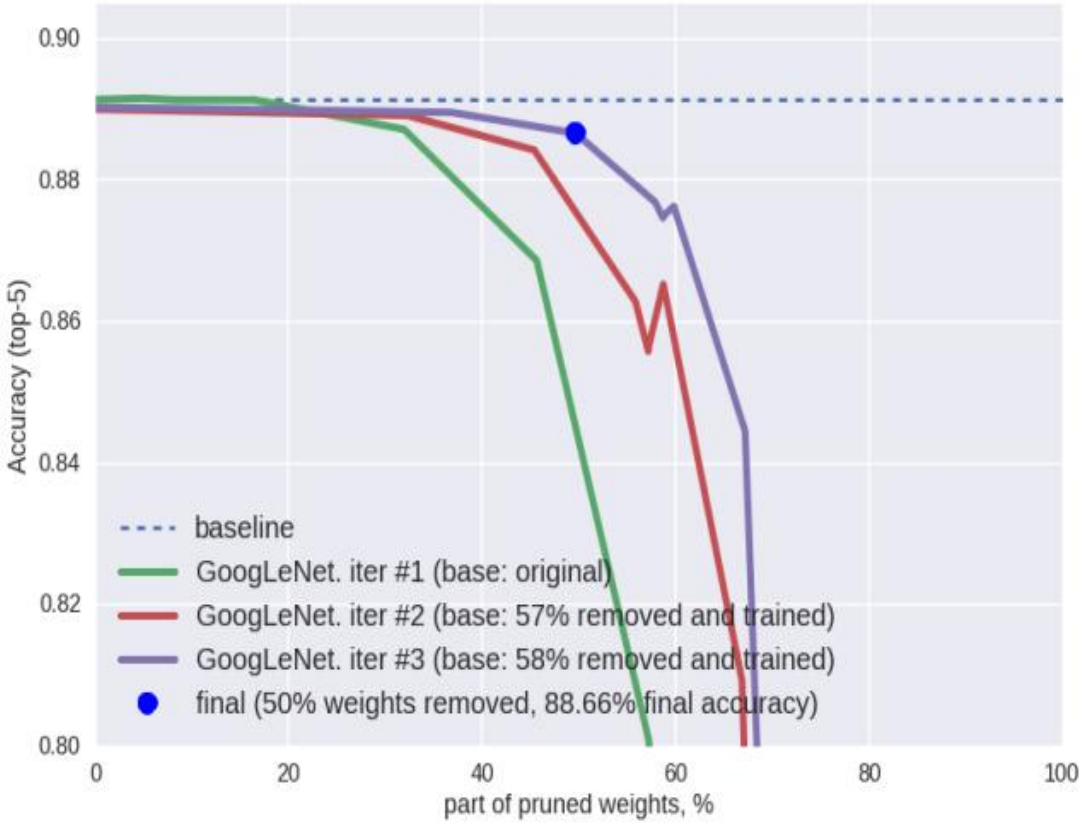


Рис. 3.2 – Спрощення мережі GoogleNet

3.4 Написання коду програми

Після того, як нейронна мережа була натренована, а все програмне забезпечення було налаштоване і встановлено, необхідно написати програму, яка використовувала б цю нейронну мережу для отримання необхідних результатів.

Так як, додаток буде розроблено для операційної системи Android буде використовуватися інтегроване середовище розробки Android Studio. Для створення програмного забезпечення під Android використовується мова Java. Крім Java можна використовувати C ++ (Android ndk), але використання C ++ не схвалюється розробниками системи. Використовувати C ++ можна тільки, якщо необхідна вкрай висока продуктивність або в інших особливих випадках. В даному випадку, нам необхідно використовувати бібліотеку tensorflow для роботи з нейронною мережею. Так як для tensorflow не розроблена API на мові Java, доведеться використовувати мову C ++. Але tensorflow - це високорівнева бібліотека, для

отримання результату необхідно просто розробити інтерфейс передачі вхідних даних і адаптер для отримання результату. Тому кількість коду на C ++ буде мінімальним. Велика частина коду програми написана для підтримки якісної взаємодії з користувачем за допомогою стандартних інтерфейсів системи Android.

Почнемо з опису головного екрану програми. В Android для опису екрана використовується компонент Activity. Activity - це компонент програми, який видає екран, і з яким користувачі можуть взаємодіяти для виконання будь-яких дій, наприклад набрати номер телефону, зробити фото, відправити лис або переглянути карту. Кожній операції присвоюється вікно для промальовування відповідного призначеного для користувача інтерфейсу. Зазвичай вікно відображається на весь екран, проте його розмір може бути менше, і воно може розміщуватися поверх інших вікон.

Головною активністю додатоку є CameraActivity, в цілому це єдина активність існуюча в данному додатку. Велику частину цього екрану займає елемент, який відображає фокус камери. Крім цього, у верхній частині екрану розміщена панель виведення результатів. Результати виводяться в трьох текстових рядках. Кожен рядок може зберігати результат, котрий складається з назви класу класифікованого об'єкта і коефіцієнта ймовірності приналежності його до цього класу. Для можливості використовувати камеру пристрою необхідно запросити у користувача дозвіл на використання його камери і доступу до внутрішньої пам'яті пристрою. Якщо на пристрої встановлений Android вище або рівній шостій версії, то необхідно запросити дозвіл під час виконання програми, як показано нижче:

```
private void requestPermission() {
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    if (shouldShowRequestPermissionRationale(PERMISSION_CAMERA) ||
shouldShowRequestPermissionRationale(PERMISSION_STORAGE)) {
      Toast.makeText(CameraActivity.this, "Camera AND storage permission are required",
Toast.LENGTH_LONG).show();
    }
    requestPermissions(new String[] {
      PERMISSION_CAMERA,
      PERMISSION_STORAGE
    }, PERMISSIONS_REQUEST);
  }
}
```

Як можна побачити з фрагмента коду вище, додаток перевіряє поточну версію операційної системи, якщо вона дорівнює або новіше ніж Android marshmallow і при цьому дозвіл на доступ до камери або внутрішнього накопичувача пристрою заборонено, то запитуємо у користувача відповідні дозволи. Для операційної системи нижче ніж Android marshmallow не потрібно запитувати дозвіл під час виконання програми. Користувач погоджується на використання цих ресурсів при установці програми.

Також ця активність виконує певні дії при переході від одного стану життєвого циклу до іншого. Наприклад, коли активність припиняється, це відбувається коли екран телефону гасне, пристрій переходить в режим очікування або фокус телефону переміщається на іншу програму. В цей момент додаток перестає виконувати всі логічні функції, так як в них не має сенсу, і користувачі не бачать результати цих обчислень. Це дозволяє заощадити ресурси телефону такі, як заряд батареї, оперативну пам'ять і процесорний час. Дані процедури зображені на фрагменті коду нижче:

```
@Override
public synchronized void onResume() {
  super.onResume();
  handlerThread = new HandlerThread("inference");
  handlerThread.start();
  handler = new Handler(handlerThread.getLooper());
}
@Override
public synchronized void onPause() {
  if (!isFinishing()) {
     finish();
  }
  handlerThread.quitSafely();
  try {
     handlerThread.join();
     handlerThread = null;
     handler = null;
  } catch (final InterruptedException e) {
     LOGGER.e(e, "Exception!");
  }  super.onPause();
}
```

Ще однією дуже важливою річчю, яку необхідно знати про activity, це те, що воно реалізує інтерфейс ImageReader.OnImageAvailableListener. Цей інтерфейс дозволяє взаємодіяти додаткам з камерою пристрію. Він викликає метод onImageAvailable (ImageReader reader), коли нове зображення можна витягти з фокуса камери. Нижче приведена блок схема взаємодії камери з інтерфейсом та додатком у цілому (рис.3.3).

Модель роботи камери представлена у вигляді конвеєра (pipeline). Це патерн ООП, який передбачає використання многопоточної розробки з метою послідовного перетворення об'єкта, виведеного як кінцевий результат. У нашому випадку підключаємося до камери, задаємо параметри зйомки і одержувача сформованого зображення. Камера - це фізичний об'єкт, тому для підвищення продуктивності і економії ресурсів пристрою доцільно все обчислення покласти на додаткові потоки. Камера - досить складний і не самий швидко реагуючий фізичний пристрій. У порівнянні з обчисленнями в пам'яті пристрою перехід камери зі стану в стан може займати цілу вічність. Тому операції з камерою будуть виконуватися поза головного потоку, а зміни стану камери будемо відловлювати за допомогою обробників подій.

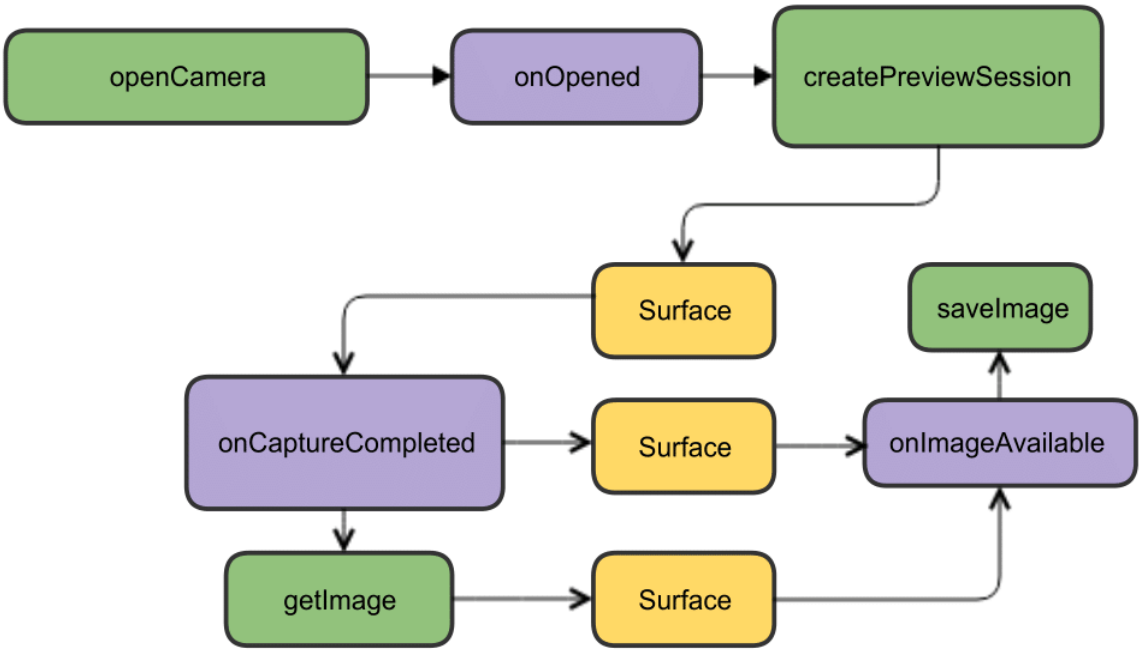


Рис. 3.3 – Схема взаємодії додатку з камерою

Так як CameraActivity є абстрактною, в ній немає реалізації методу інтерфейсу OnImageAvailableListener. Цей метод реалізовано в активності, яка наслюдується від CameraActivity, про яку мова піде далі.

Основні функції, які виконують Camera Activity були перераховані. Всю решту логіки активність передає всередину фрагмента, який розміщений в даному Activity. Фрагмент (клас Fragment) представляє поведінку або частину користувацького інтерфейсу в операції (клас Activity). Розробник може об'єднати кілька фрагментів в одну операцію для побудови багатопанельного призначеного для користувача інтерфейсу і повторного використання фрагмента в декількох операціях. Фрагмент можна розглядати як модульну частину операції. Така частина має свій життєвий цикл і самостійно обробляє події введення. Крім того, її можна додати або видалити безпосередньо під час виконання операції. Це щось на зразок вкладеної операції, яку можна багаторазово використовувати в різних операціях. Нижче наведена частина коду, яка ініціалізує фрагмент.

```
protected void setFragment() {
final Fragment fragment = CameraConnectionFragment.newInstance(
    new CameraConnectionFragment.ConnectionCallback() {
      @Override
      public void onPreviewSizeChosen(final Size size, final int rotation) {
        CameraActivity.this.onPreviewSizeChosen(size, rotation);
      }
    },
    this, getLayoutId(), getDesiredPreviewFrameSize());

getFragmentManager()
    .beginTransaction()
    .replace(R.id.container, fragment)
    .commit();
}
```

Фрагмент, котрий знаходиться всередині вищезгаданого Activity називається CameraConnectionFragment. Цей фрагмент бере на себе всю логіку відповідальну за роботу камери, взаємозв'язок з камерою, перетворення зображення з камери у необхідний формат, відображення вхідного потоку інформації з камери безпосередньо на екран пристрою. Фрагмент обчислює розміри вікна, в якому буде відображений фокус камери. Він перетворює орієнтацію екрану до JPEG формату.

Більш високорівневу логіку програмної реалізації бере на себе підклас Camera Activity, який називається ClassifierActivity. Дана активність бере на себе логіку по обчисленню

результату. Основна логіка всього додатку зосереджена саме тут. У цій активності реалізований вищезгаданий метод onImageAvailable. В якому взяте з камери зображення перетворюється і передається в адаптер, який передає вхідні дані бібліотеці tensorflow, а та в свою чергу, передає їх на вхід нейронної мережі, отримує результат і повертає їх за допомогою необхідного інтерфейсу. Дана процедура вказана на фрагменті коду нижче.

```
@Override
public void onImageAvailable(final ImageReader reader) {
  Image image = null;

  try {
    image = reader.acquireLatestImage();
    if (image == null) {
      return;
    }
    if (computing) {
      image.close();
      return;
    }
    computing = true;
    Trace.beginSection("imageAvailable");
    final Plane[] planes = image.getPlanes();
    fillBytes(planes, yuvBytes);
    final int yRowStride = planes[0].getRowStride();
    final int uvRowStride = planes[1].getRowStride();
    final int uvPixelStride = planes[1].getPixelStride();
    ImageUtils.convertYUV420ToARGB8888(
      yuvBytes[0], yuvBytes[1], yuvBytes[2], rgbBytes,previewWidth, previewHeight,
      yRowStride,uvRowStride, uvPixelStride,false);
    image.close();
  } catch (final Exception e) {
    if (image != null) {
      image.close();
    }
    Trace.endSection();
```

```
            return;
          }
          rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);
          final Canvas canvas = new Canvas(croppedBitmap);
          canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);
          if (SAVE_PREVIEW_BITMAP) {
            ImageUtils.saveBitmap(croppedBitmap);
          }
          runInBackground(
            new Runnable() {
              @Override
              public void run() {
                final long startTime = SystemClock.uptimeMillis();
                final List < Classifier.Recognition > results =
classifier.recognizeImage(croppedBitmap);
                lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

                cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
                resultsView.setResults(results);
                requestRender();
                computing = false;
              }
            });
          Trace.endSection();
        }
```

Як видно на фрагменті коду вище, основна логіка отримання результату відбувається в об'єкті classifier класу TensorFlowImageClassifier, для ініціалізації якого використовуються фабричний метод. У якості параметрів в цей фабричний метод передаються такі дані як: шлях до файлу моделі нейронної мережі, кількість класів, шлях до файлу з іменами класів, розмір вхідного зображення, і інші менш цікаві для нас дані. Саме клас TensorFlowImageClassifier виступає в якості вищезгаданого адаптера бібліотеки tensorflow, передає на вхід всі необхідні дані. Передача даних відбувається за допомогою інтерфейсу TensorFlowInferenceInterface.

3.5 Результати роботи програми

Після того, як код був написаний і протестований, можна робити білд додатку. Білд проекту робиться за допомогою збирача проектів Bazel. Установка цього збирача описувалася вище. Існує можливість установки цього збирача на декількох операційних системах (OS X, Linux, Windows). Для мене приоритетніше було б працювати на Windows, але для Windows реалізація Bazel знаходиться в експериментальному стані. І не вдалося налаштувати збирач на цій операційній системі. Тому використано персональний комп'ютер з операційною системою OS X.

Основна робота проведена, код написаний, додаток скомпільовано. Тепер необхідно провести перевірку точності роботи системи. Перш ніж почати аналізувати результати роботи мережі, натренованої за допомогою датасета, хотілося б провести невеликий аналіз роботи інших нейронних мереж і inceptionV3. Почнемо з мережі AlexNet. Розробники даної нейронної мережі натренували глибоку згорточну нейронну мережу за допомогою півтора мільйона зображень з високою роздільною здатністю для конкурсу ImageNet 2010 року для розрізнення 1000 різних класів. Їм вдалося досягти високих результатів. Коефіцієнт помилок 37,5 відсотків і 17,0 відсотків, що значно краще ніж в попередніх змаганнях. Нейронна мережа має 6 мільйонів параметрів і 650000 нейронів, складається з 5 згорточних шарів. Нижче наведено зображення з результатами роботи нейронної мережі на декількох зображеннях.

Як видно, для більшості зображень мережа успішно змогла класифікувати об'єкти. Але для деяких зображень все ж дала не зовсім точні результати. Далі хотілося б навести приклади роботи нейронної мережі inceptionV3. Саме цю нейронну мережу використано в даній роботі, натренувавши на свої тренувальні дані. Для отримання результатів роботи цієї нейронної мережі буде використана модель розроблена для того ж конкурсу imaginet. Цю мережу просто підмінено в розроблений додаток замість тієї, яку треновано. Далі показано зображення, на якому знаходяться результати роботи цієї нейронної мережі для тих же зображень, які були використані для перевірки мережі AlexNet (рис.3.4).

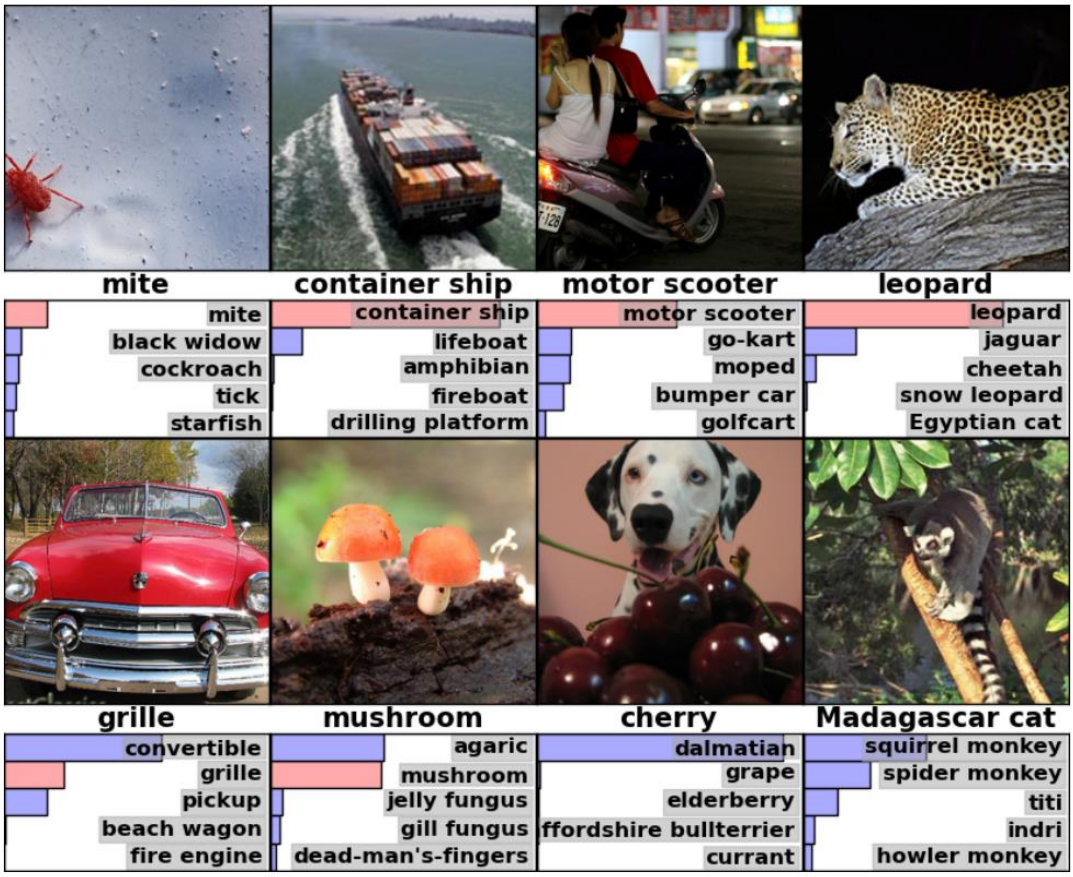


Рис. 3.4 – Результат роботи мережі AlexNet

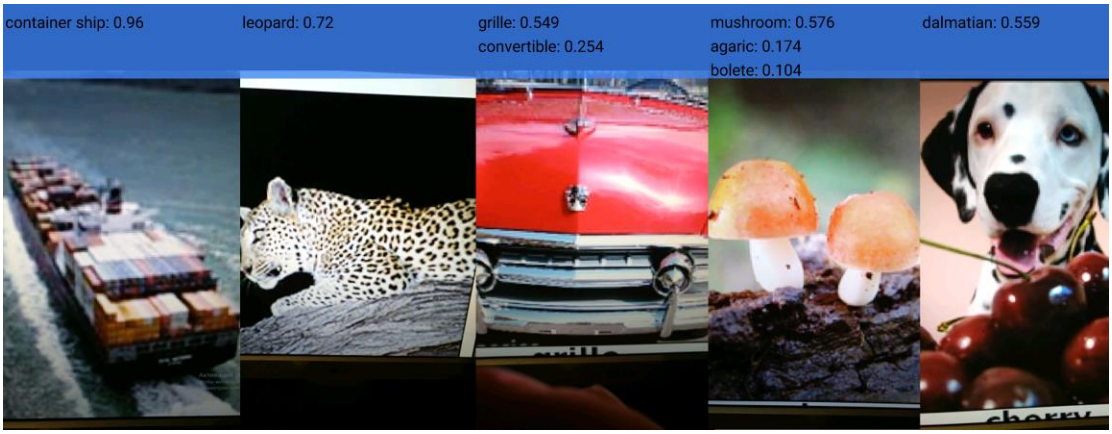


Рис. 3.5 – Результат роботи мережі InceptionV3

З вище показаних зображень можна зробити висновок, що нейронні мережі показують приблизно однаково хороші результати. В деяких випадках inceptionV3 більш точна у своїх свідченнях. Але варто врахувати, що inceptionV3 працювала на мобільному пристрої. У той час, як AlexNet виробляв обчислення на потужному комп'ютері. Тому, результати inceptionV3 вище ніж AlexNet.

Далі необхідно проаналізувати результати роботи програми на натренованій моделі inceptionV3. Нижче показаний результат роботи програми при класифікації класу roses (рис.3.6).

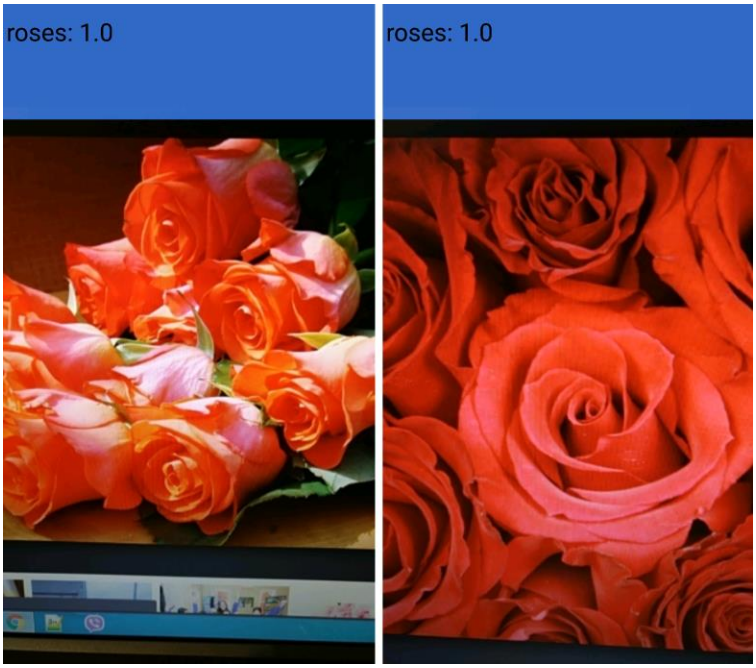


Рис. 3.6 - Результат роботи натренованої мережі InceptionV3

Як видно з рисунку, нейронна мережа продемонструвала неймовірну точність. Це дуже несподіваний результат, так як нейронні мережі в дуже рідкісних випадках видають результат рівний одиниці. Звичайно швидше за все результат був округлено, але тим неменш, це означає, що він був максимально наближений до одиниці. Нагадаємо, що результатом є коефіцієнт ймовірності приналежності знаходження на зображенні об'єкта до того чи іншого класу. Чим ближче до одиниці значення, тим вище ступінь ймовірності приналежності об'єкта до класу. Ще одним дуже важливим аспектом є те, що зображення, яке знаходиться у фокусі камери не знаходилося в тренувальному наборі. Зазвичай такі високі результати досягаються, коли зображення взято з тренувального набору. Це неправильно, так як, очевидно, що для такого зображення буде високий результат.

Наступним класом, для якого буде проводитися перевірка є sunflowers. Нижче показаний результат класифікації додатком соняшнику.

Рис. 3.7 - Результат розпізнавання соняшника

Як бачимо, нейронна мережа досить успішно впоралася зі своїм завданням за визначенням соняшника на зображенні. Слід згадати, що зображення соняшника на рисунку 3.7, взято не з тренувального датасета. Проте, результат досить високий. Ще хотілося б звернути увагу на те, що нейронна мережа не дала більше ніяких можливих результатів, це теж говорить про високу якість роботи нейронної мережі.

Настав час перейти до найцікавішого етапу перевірки реалізованого додатку. Наступним кроком необхідно перевірити якість розпізнавання об'єкта чашка. Тренувальні дані для цього об'єкта, як згадувалося вище, збиралися не в мережі інтернет, а за допомогою знімків зроблених на власну фотокамеру. На рисунку 3.8 показаний результат класифікації додатком об'єкта «dntl cup».

Результат нижче теж досить високий. При цьому зображення зроблено з різних ракурсів, так як тестування проводилося з реальним об'єктом, а не з зображенням з екрану монітора. Як не дивно, з різних ракурсів результат був приблизно однаковий і вище ніж 0,9. Чашки, над якою проводився експеримент, теж не було в числі тих, які перебували в тренувальному наборі даних. Це свідчить про те, що нейронна мережа спроектована надзвичайно вдало. Тренувальний dataset зібраний правильно. Якщо проводити порівняльну характеристику даних, які були зібрані в інтернеті та тих, які були зроблені на власну камеру, то великої різниці не можна виявити. Але не можна заперечувати факт того, що для об'єктів натренованих за допомогою даних з інтернету - коефіцієнт точності, хоч і не багато, але вище.

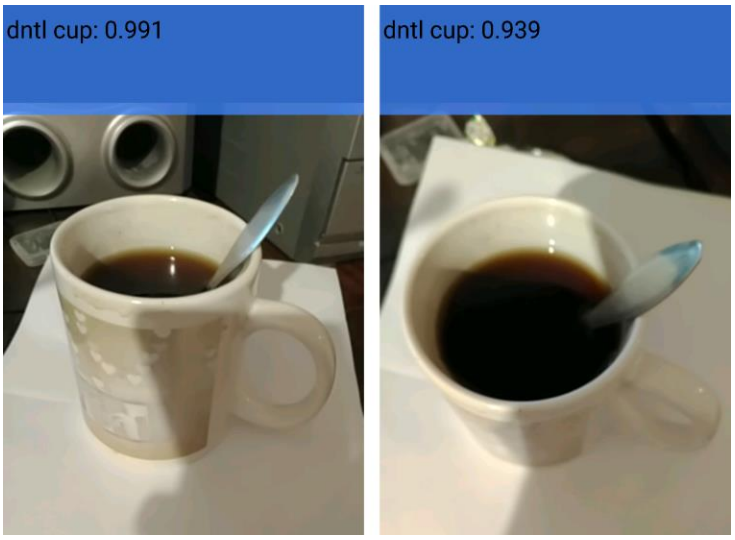


Рис. 3.8 - Результат розпізнавання чашки

Що стосується ситуацій, при яких у фокусі камери не перебуває жодний з об'єктів, які належать до класифікуючих класів. Якщо у фокусі камери не знаходиться жодного об'єкта, який можна віднести до того чи іншого класу, система видає приблизно однакові коефіцієнт для кожного з трьох класів, який приблизно дорівнює 0,5. Це звичайно дуже округлені дані, так як здебільшого все залежить від того, що безпосередньо знаходиться у фокусі камери. Різні об'єкти можуть мати схожі форми і характеристики, тому коефіцієнт може змінюватися. Так, наприклад, зображенню нарциса система віднесе до класів троянда або соняшник. А ось, наприклад, відро з великим коефіцієнтом буде ставитися до класу чашка. Ця ситуація зображена на рисунку 3.9.

Рис. 3.9 - Результат розпізнавання натренованої мережі InceptionV3

Нижче наведено приклад ситуації, коли у фокусі камери не знаходиться жодного об'єкта, який можна було хоча б віддалено віднести до того чи іншого класу (рис. 3.10).

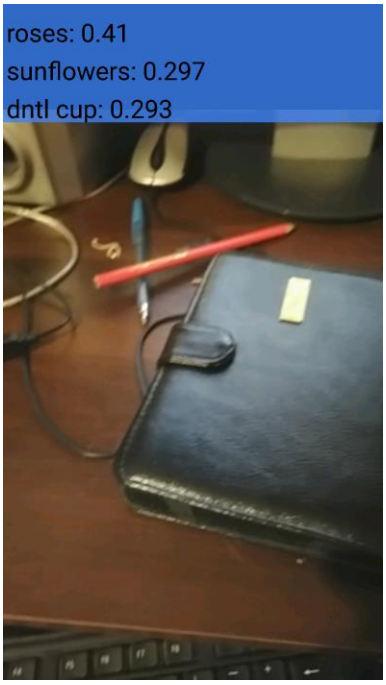


Рис. 3.10 - Результат розпізнавання натренованої мережі InceptionV3

На рисунку вище зображено результат роботи програми, коли у фокусі камери пристрою знаходиться робочий стіл. Отже на робочому столі не знаходиться жодного об'єкта з списку класів. Як бачимо, нейронна мережа віддала свою перевагу в розмірі 0,41 тому, що у фокусі камери знаходиться троянда і приблизно по 0,3 на інші класи. Це свідчить про те, що при маленькому списку класів можливі такі помилкові результати. Але якби з додатком потрібно було б повідомити користувача про те, що в фокусі камери знаходиться той чи інший об'єкт, слід було б встановити кордон коефіцієнта, приблизно в 0,9. Мається на увазі, що якщо коефіцієнт власності об'єкта дорівнює або більше ніж 0,9 - можна сміливо стверджувати, що в фокусі камери знаходиться об'єкт, який належить вказаному класу.

3.6 Висновки до розділу

У розділі було розроблено додаток для визначення об’єктів на зображенні. Була натренована нейронна мережа для класифікації більш ніж 1000 образів. Для тренування були використані данні з ресурсу ImageNet. В результаті додаток успішно класифікує об’єкти на зображеннях.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКИ В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера, на якому буде виконуватися розробка.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

4.1.1 Правові та організаційні основи охорони праці

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави, і особистої відповідальності працівників.

4.1.2 Організаційно-технічні заходи з безпеки праці

В організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 НПАОП 0.00-4.12-05 [29].

Також впроваджені організаційні заходи з пожежної безпеки - навчання і перевірку знань відповідно до вимог Типового положення про інструктажі, спеціальне навчання та

перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 29.09.2003 N 368, зареєстрованого в Міністерстві юстиції України 11.12.2003 за N 1148/8469 НАПБ Б.02.005-2003[32]. Обов'язковими вимогами враховане наступне:

1. не слід допускати до роботи осіб, що в установленому порядку не пройшли навчання, інструктаж та перевірку знань з охорони праці, пожежної безпеки та цих Правил.

2. на підприємстві/організації, де експлуатуються ЕОМ з відео дисплейними терміналами (ВДТ) і периферійними пристроями (ПП), розробляється інструкція з охорони праці відповідно до Положення про розробку інструкцій з охорони праці [30].

3. ознайомлення з правилами безпеки праці, одержання відповідних інструктажів засвідчується у журналі інструктажів.

4. перед допуском до самостійної роботи кожен працівник має право на навчання з питань охорони праці і роботодавець зобов'язаний, і проводить таке навчання у вигляді двох інструктажів з питань охорони праці: вступного та первинного.

4.2 Аналіз стану умов праці

Робота над проектом проходитиме в приміщенні багатоквартирного будинку. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 5.1.

Таблиця 4.1 – Розміри приміщення

Найменування	**Значення**
Довжина, м	5
Ширина, м	5
Висота, м	3
Площа, $м^2$	25
Об'єм, $м^3$	75

Згідно з [14] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [26] (табл. 5.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Приміщення кабінету знаходиться на другому поверсі трьох поверхової будівлі і має об’єм 78 м^3, площу – 18 м^2. У цьому кабінеті обладнано три місця праці, з яких два укомплектовані ПК.

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум в лабораторії знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

4.2.3 Навантаження та напруженість процесу праці

Під час виконання випускної роботи:

за фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені в [26].

Роботу за дипломним проектом визнано, такою, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви - для розробників програм тривалістю 15 хв. через кожну годину роботи;

4.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 5.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00.-1.28-10 [34], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Основними робочими характеристиками персонального комп'ютера є:

1. робоча напруга U=+220В +-5%;
2. робочий струм I=2А;
3. споживана потужність P=350 Вт.

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[14]
- підвищений рівень шуму на робочому місці	-//-	2	[13]
- підвищений рівень вібрації	-//-	2	[14] [27]
- підвищена або знижена вологість повітря	-//-	2	[14]
- підвищена або знижена рухливість повітря	-//-	1	[14]
- підвищений рівень іонізуючого випромінення в робочій зоні	-//-	2	[14] [24]
- підвищений рівень електромагнітного випромінення	-//-	2	[24]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[25] [14]
- підвищений рівень статичної електрики	-//-	2	[25]
- підвищена напруженість електричного поля	-//-	2	[24]
- підвищена напруженість магнітного поля	-//-	2	[24]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[11]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[11]
- підвищена яскравість світла	порушення умов праці (організації місця праці- налагодження моніторів)	1	[12]
- понижена контрастність	-//-	1	[12]

Продовження табл. 4.3

психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[26] [12]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці - сидіння користувача) та організації робочого часу - безпервна робота)	2	[26] [12]

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [26].

4.3.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі і важкогорючі) не надається технічно можливим. Тому проектом передбачаються способи і засоби запобігання утворення (або внесення) в горюче середовище джерел запалювання, таких як:

1. застосування електроустаткування, відповідної пожежонебезпечної і вибухонебезпечної зонам відповідно до ПУЕ;
2. застосування в конструкції швидкодійних засобів захисного відключення можливих джерел запалення;
3. виключення можливості появи іскрового розряду в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення.

Згідно [18] таке приміщення, площею 25 м^2, відноситься до категорії "В" (пожежонебезпечної) та для протипожежного захисту в ньому проектом передбачено устаткування автоматичною пожежною сигналізацією із застосуванням датчиків-сповіщувачів РІД-1 (сповіщувач димовий ізоляційний) в кількості 1 шт., і застосуванням первинних засобів пожежогасіння.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до [32] відносяться до пожежонебезпечної зони класу П-IIа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та

важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється II ступень вогнестійкості.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол [24].

4.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [27] і наведені в табл. 4.4:

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [27]. Рівні позитивних і негативних іонів у повітрі мають відповідати [27].

Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.4.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає [25]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 5 = 25$ м^2,

$S = 1/8 \cdot 25 = 3{,}125$ м^2.

Приймаємо 2 вікна площею S=1,6 м^2 кожне.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \quad (4.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (А.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2{,}0 \quad (4.3)$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп'ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [13]. У залах опрацювання інформації та комп'ютерного набору рівні шуму не повинні перевищувати 65 дБА.

Віброізоляція можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [27].

4.4.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при V приміщення > 40 м3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);
- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря.

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;
- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;
- не тягнути за мережевий кабель, щоб витягти вилку з розетки;
- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

- не залишати включені електроприлади без нагляду;

- не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

4.5.1 Вимоги безпеки при надзвичайних ситуаціях:

1) При раптовому припиненні подачі електричної енергії вимкнути всі пристрої ПК в такій послідовності: периферійні пристрої, ВДТ, системний блок, стабілізатор (або блок безперервного живлення). Витягнути вилки з розеток. При наявності ознак горіння (дим, запах горілого) необхідно вимкнути всі пристрої ПК, знайти місце загоряння і виконати всі можливі заходи для його ліквідації, попередивши терміново про це керівництво.

2) При замиканні, перевантаженні електричного струму на електричному обладнанні, внаслідок ураження грозової блискавки та ймовірної небезпеки ураженням електричним струмом, приймають наступне:

- попередження замикання здійснюється правильним вибором, монтажем експлуатації мереж;

- застосування захисту схем у вигляді швидкодіючих реле, а також вимикачів, плавких запобіжників.

Також застосовують різні **електричні захисні засоби від ураження струмом**:

а) Ізолюючі - ізолюють людини від струмоведучих або заземлених частин, а так-же від землі. Вони діляться на основні та додаткові.

б) Основні - володіють ізоляцією, здатної довго витримувати робоче напругу електроустановки і тому ними дозволяється стосуватися струмоведучих частин, знаходячи-трудящих під напругою.

в) Запобіжні - володіють ізоляцією нездатною витримати робоча напруга електроустановки, і тому вони не можуть самостійно захищати людину від ураження струмом під цим напругою.

4.5.2 Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [19], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача η_c.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \tag{4.4}$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (А.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом} \tag{4.5}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \tag{4.6}$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м} \tag{4.7}$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м} \tag{4.8}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (5.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (4.9)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (5.6):

$$t = h_в + \frac{l_в}{2}, \quad (4.10)$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0,8 + \frac{3}{2} = 2,3$ м

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом} \quad (4.11)$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.12)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.13)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_с = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.14)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05 \cdot 3 \cdot (16-1) \approx 48 \text{ м} \quad (4.15)$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_с} \cdot \ln \frac{2 \cdot l_с^{\,2}}{d_{см} \cdot h_г}, \quad (4.20)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

$l_с$ - довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом} \quad (4.21)$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_с$. відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.22)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д \quad (4.23)$$

3) При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

- іскри і дуги коротких замикань;

- перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;

- іскри при розмиканні і розмиканні ланцюгів;

- розряди статичної електрики;

- необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

4.6 Охорона навколишнього природного середовища

4.6.1 Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме, процес виконання якої впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на знешкодження, утилізацію, тощо в ІТ галузі.

В процесі діяльності виявлення проблем при роботі з даними з соціальних мереж та підвищення точності емоційної класифікації виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

— Відпрацьовані люмінесцентні лампи - І клас небезпеки

— Акумулятор для джерел безперебійного живлення -ІІІ клас небезпеки

— Макулатура - IV клас небезпеки

— Побутові відходи - IV клас небезпеки

4.6.2 Вимоги до збору, пакування та розміщення відходів ІТ галузі

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм [41].

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів

відповідно до затвердженої схеми їх розміщення. Зазначені для зберігання відходів місця чи об'єкти повинні використовуватися лише для заявлених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- Відходи I класу небезпеки зберігаються в герметичній тарі (сталеві бочки, контейнери). У міру наповнення тару з відходами закривають герметично сталевий кришкою;

- Відходи II класу небезпеки в залежності від агрегатного стану зберігаються в поліетиленових мішках, бочках, сховищах та інших видах тари, яка запобігає поширенню шкідливих речовин;

- Відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;

- Відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення;

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) і ін.

Особливий контроль наділяється збору і зберіганню відпрацьованих ртутьвмісних ламп (енергоощадних) як відходам I класу небезпеки, що збираються і обов’язково передаються на утилізацію підприємствам, що мають ліцензію на поводження з такими небезпечними відходами.

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Вимоги безпеки при поводженні з відходами:

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Наведено перелік деяких відходів, які передаються на утилізацію організаціям, які мають ліцензію на поводження з відходами як вторинної сировини:

- Макулатура;

- Матеріали пакувальні вторинні

Відвантаження таких відходів здійснюється відповідно до договору (контракту).

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

4.6.3 Визначення впливу та заходів щодо поводження з відходами ІТ галузі

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів.

Відомості про місце утворення та місце розташування відходів зазначаються на «План схемі місці розміщення відходів організації / виробництва».

4.7 Висновки до розділу

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Було визначено загальні дані з охорони навколишнього середовища. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

Основною перевагою застосування нейронних мереж є можливість вирішувати різні неформалізовані завдання. При цьому можна дуже просто моделювати різні ситуації, подаючи на вхід мережі різні дані і оцінюючи виданий мережею результат.

В ході застосування нейронних мереж відзначений істотний недолік: складність розуміння процесу одержання мережею результату. Першим кроком до вирішення цієї проблеми є розробка нової технології, яка дає змогу генерувати опис процесу рішення задачі нейронною мережею. Використовуючи таблицю експериментальних даних, що описують предметну область, можна буде отримати явний алгоритм вирішення поставленого завдання. Потрібні подальші експерименти з дослідження алгоритмів навчання, вибору початкового уявлення і впровадженню в архітектуру НМ обліку властивостей зображення.

Хоча існує безліч і програмних засобів моделювання нейромереж досі найчастіше використовуються моделі створені на мовах високого рівня під кожен конкретний експеримент, що приводить до невиправдано високих витрат і низькій швидкодії систем. Також часто використовуються свої власні бази даних, що істотно ускладнює процес порівняння результатів різних експериментів, хоча у відкритому доступі є спеціалізовані бази. Практично повністю відсутні експерименти з апаратними реалізаціями нейромережевих методів біометричного розпізнавання.

Можливість запуску нейронної мережі на мобільному пристрої свідчить про швидкий розвиток і великі подальші перспективи технології. Тепер не потрібно мати потужний обчислювальний центр, щоб користуватися перевагами НМ. І у майбутньому, нейронні мережі зможуть працювати швидше і точніше навіть на мобільному пристрої.

Розроблено додаток для визначення об’єктів на зображенні. Була натренована нейронна мережа для класифікації більш ніж 1000 образів. Для тренування були використані данні з ресурсу ImageNet. В результаті додаток успішно класифікує об’єкти на зображеннях.

Були визначені параметри і певні характеристики приміщення для роботи. Приведені рекомендації щодо організації робочого місця, а також приведена важлива інформацію щодо пожежної та електробезпеки. Наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері. Приведені рекомендації щодо охорони навколишнього середовища.

ПЕРЕЛІК ПОСИЛАНЬ

1. Васильєв В.М., Гуров І.П., Потапов А.С. Сучасна відеоінформатика: проблеми і перспективи / 2012, № 11, с. 5-15.
2. Абламейко С.В., Лагуновскій Д.М. Обробка зображень: технологія, способи, застосування / Білорусь, - Мінськ: Ін-т техн. Кібернетики НАН Білорусі, 2000., - 304 с.
3. Блаттер К. Вейвлет-аналіз. Основи теорії / - М. - Техносфера. - 2006, - 279 с.
4. Васильєв В.Н, Гуров І.П., Потапов А.С. математичні методи і алгоритмічне забезпечення аналізу і розпізнавання зображень в інформаційно-телекомунікаційних системах / Всеросійський конкурсний відбір обзорно аналітичних статей по пріоритетному напрямку «Інформаційно-телекомунікаційні системи», - 2008, - 46 с.
5. Тропченко А. Ю. Методы вторичной обработки и распознавания изображений / Тропченко А. Ю., Тропченко А.А., 2015, - 45 с.
6. Востріков А.С., Пустовий Н.В. Цифрова обробка зображень в інформаційних системах / Учебник НГТУ, Новосибірськ 2002.
7. Галушкин А. І., Томашевич Д. С., Томашевич М. С. Методи реалізації інваріантності до афінних перетворень двовимірних зображень / Додаток до журналу «Інформаційні технології», - 2001, - №1, с. 1-19.
8. Галушкин А.І. Деякі історичні аспекти розвитку елементної бази обчислювальних систем з масовим паралелізмом (80- і 90-ті роки) / Нейрокомп'ютер, №1, 2000, с. 68-82.
9. Глазунов О. Комп'ютерне розпізнавання людських осіб / Відкриті системи, 2000., №3.
10. Гонсалес Р., Вудс Р. Цифрова обробка зображень / Пер. з англ Москва.- Техносфера. - 2006, -1072 с.
11. Горелик А.Л., Скрипкін В.А. Методи розпізнавання / - М: Вища школа, 1984, - 208 с.
12. Кухарев Г.А. Біометричні системи: Методи і засоби ідентифікації особистості людини / - СПб., Політехніка, 2003 - 240 с.
13. Кухарев Г. А. Методи обробки і розпізнавання зображень обличь в задачах біометрії / Г. А. Кухарев, Е. І. Каменська, Ю. М. Матвєєв, Н. Л. Щеголева; під ред. М. В. Хитрова. - СПб .: Політехніка 2013, - 388 с.
14. Осовскій С. Нейронні мережі для обробки інформації / - М .: Фінанси і статистика, 2002
15. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем/ А.М. Вендров. - М.: Финансы и статистика, 1998.–176 с.
16. Маклаков, С.В. BPWin и ERWin. Case-средства разработки информационных систем/ С.В.Маклаков-М.: ДИАЛОГ–МИФИ, 1999.–256с.

17. Орлов, С.А. Технологии разработки программного обеспечения/ С.А. Орлов–СПб.: Питер, 2002.–464 с.
18. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс / Гарсиа-Молина Г, Ульман Дж, Уидом Дж. — М.: "Вильямс", 2003. – 229 с.
19. Дейт. К. Дж. Введение в системы баз данных / К. Дж. Дейт. — "Вильямс", 2001. – 426 с.
20. Харрингтон Д. Л Проектирование реляционных баз данных. Просто и доступно / Д. Л. Харрингтон. – М.: ЛОРИ, 2000. – 277 с.
21. Коннолли Т. М, Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. М. Коннолли, К. Бегг. – М.: Издательский дом "Вильямс", 2003. – 261 с.
22. Калянов Г. Н. CASE. Структурный системный анализ (автоматизация и применение) / Г. Н. Калянов. – М.: "Лори", 2006. – 175 с.
23. Черемных, С.В. Структурный анализ систем: IDEF-технологии. /С.В. Черемных, И.О.Семенов, В.С. Ручкин-М.: Финансы и статистика, 2003.–208 с.
24. ГОСТ 12.1.044-89 ССБТ. Пожежовибухонебезпека речовин і матеріалів. Номенклатура показників і методи їх визначенняя
25. ДБН В.2.5-28:2015 Природнє і штучне освітлення
26. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин
27. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку
28. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих
29. НПАОП 0.00-4.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці
30. НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці
31. НПАОП 0.00-6.03-93 Порядок опрацювання та затвердження власником нормативних актів про охорону праці
32. НАПБ Б.03.002-2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою
33. НПАОП 40.1-1.01-97 Правила безопасной эксплуатации электроустановок
34. НПАОП 40.1-1.32-01 Правила устройства электроустановок. Электрооборудование специальных установок
35. ДСН 3.3.6.039-99 Санітарні норми виробничої загальної та локальної вібрації
36. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, льтразвуку та інфразвуку»
37. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування

38. ГОСТ 12.1.006-84 ССБТ. Електромагнітні поля радіочастот. Загальні вимоги безпеки. Допустимі рівні на робочих місцях і вимоги до проведення контролю
39. ГОСТ 12.1.030-81 ССБТ. Електробезпечність. Захисне заземлення. Занулення
40. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин
41. Державні санітарні правила і норми (ДСанПіН 2.2.7.029) [Електронний ресурс] / LIGA:ZAKON – Режим доступу: www.URL: http://zakon.sop.com.ua/regulations/10637/478449/

ДОДАТОК А

Лістинг програми

```
public abstract class CameraActivity extends Activity implements OnImageAvailableListener {
  private static final Logger LOGGER = new Logger();

  private static final int PERMISSIONS_REQUEST = 1;

  private static final String PERMISSION_CAMERA = Manifest.permission.CAMERA;
  private static final String PERMISSION_STORAGE =
Manifest.permission.WRITE_EXTERNAL_STORAGE;

  private boolean debug = false;

  private Handler handler;
  private HandlerThread handlerThread;

  @Override
  protected void onCreate(final Bundle savedInstanceState) {
   LOGGER.d("onCreate " + this);
   super.onCreate(null);
   getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

   setContentView(R.layout.activity_camera);

   if (hasPermission()) {
    setFragment();
   } else {
    requestPermission();
   }
  }

  @Override
```

```
public synchronized void onStart() {
 LOGGER.d("onStart " + this);
 super.onStart();
}

@Override
public synchronized void onResume() {
 LOGGER.d("onResume " + this);
 super.onResume();

 handlerThread = new HandlerThread("inference");
 handlerThread.start();
 handler = new Handler(handlerThread.getLooper());
}

@Override
public synchronized void onPause() {
 LOGGER.d("onPause " + this);

 if (!isFinishing()) {
  LOGGER.d("Requesting finish");
  finish();
 }

 handlerThread.quitSafely();
 try {
  handlerThread.join();
  handlerThread = null;
  handler = null;
 } catch (final InterruptedException e) {
  LOGGER.e(e, "Exception!");
 }

 super.onPause();
}
```

```
@Override
public synchronized void onStop() {
 LOGGER.d("onStop " + this);
 super.onStop();
}

@Override
public synchronized void onDestroy() {
 LOGGER.d("onDestroy " + this);
 super.onDestroy();
}

protected synchronized void runInBackground(final Runnable r) {
 if (handler != null) {
  handler.post(r);
 }
}

@Override
public void onRequestPermissionsResult(
   final int requestCode, final String[] permissions, final int[] grantResults) {
 switch (requestCode) {
  case PERMISSIONS_REQUEST: {
   if (grantResults.length > 0
       && grantResults[0] == PackageManager.PERMISSION_GRANTED
       && grantResults[1] == PackageManager.PERMISSION_GRANTED) {
    setFragment();
   } else {
    requestPermission();
   }
  }
 }
}
```

```
    private boolean hasPermission() {
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      return checkSelfPermission(PERMISSION_CAMERA) ==
PackageManager.PERMISSION_GRANTED && checkSelfPermission(PERMISSION_STORAGE)
== PackageManager.PERMISSION_GRANTED;
     } else {
      return true;
     }
    }

    private void requestPermission() {
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      if (shouldShowRequestPermissionRationale(PERMISSION_CAMERA) ||
shouldShowRequestPermissionRationale(PERMISSION_STORAGE)) {
       Toast.makeText(CameraActivity.this, "Camera AND storage permission are required for this
demo", Toast.LENGTH_LONG).show();
      }
      requestPermissions(new String[] {PERMISSION_CAMERA, PERMISSION_STORAGE},
PERMISSIONS_REQUEST);
     }
    }

    protected void setFragment() {
     final Fragment fragment = CameraConnectionFragment.newInstance(
       new CameraConnectionFragment.ConnectionCallback(){
        @Override
        public void onPreviewSizeChosen(final Size size, final int rotation) {
         CameraActivity.this.onPreviewSizeChosen(size, rotation);
        }
       },
       this, getLayoutId(), getDesiredPreviewFrameSize());

     getFragmentManager()
       .beginTransaction()
       .replace(R.id.container, fragment)
```

```
      .commit();
  }

  protected void fillBytes(final Plane[] planes, final byte[][] yuvBytes) {
   // Because of the variable row stride it's not possible to know in
   // advance the actual necessary dimensions of the yuv planes.
   for (int i = 0; i < planes.length; ++i) {
    final ByteBuffer buffer = planes[i].getBuffer();
    if (yuvBytes[i] == null) {
     LOGGER.d("Initializing buffer %d at size %d", i, buffer.capacity());
     yuvBytes[i] = new byte[buffer.capacity()];
    }
    buffer.get(yuvBytes[i]);
   }
  }

  public boolean isDebug() {
   return debug;
  }

  public void requestRender() {
   final OverlayView overlay = (OverlayView) findViewById(R.id.debug_overlay);
   if (overlay != null) {
    overlay.postInvalidate();
   }
  }

  public void addCallback(final OverlayView.DrawCallback callback) {
   final OverlayView overlay = (OverlayView) findViewById(R.id.debug_overlay);
   if (overlay != null) {
    overlay.addCallback(callback);
   }
  }

  public void onSetDebug(final boolean debug) {}
```

```
    @Override
    public boolean onKeyDown(final int keyCode, final KeyEvent event) {
      if (keyCode == KeyEvent.KEYCODE_VOLUME_DOWN || keyCode ==
KeyEvent.KEYCODE_VOLUME_UP) {
        debug = !debug;
        requestRender();
        onSetDebug(debug);
        return true;
      }
      return super.onKeyDown(keyCode, event);
    }

    protected abstract void onPreviewSizeChosen(final Size size, final int rotation);
    protected abstract int getLayoutId();
    protected abstract int getDesiredPreviewFrameSize();
  }


  public class CameraConnectionFragment extends Fragment {
    private static final Logger LOGGER = new Logger();

    /**
     * The camera preview size will be chosen to be the smallest frame by pixel size capable of
     * containing a DESIRED_SIZE x DESIRED_SIZE square.
     */
    private static final int MINIMUM_PREVIEW_SIZE = 320;

    /**
     * Conversion from screen rotation to JPEG orientation.
     */
    private static final SparseIntArray ORIENTATIONS = new SparseIntArray();
    private static final String FRAGMENT_DIALOG = "dialog";

    static {
```

```
 ORIENTATIONS.append(Surface.ROTATION_0, 90);
 ORIENTATIONS.append(Surface.ROTATION_90, 0);
 ORIENTATIONS.append(Surface.ROTATION_180, 270);
 ORIENTATIONS.append(Surface.ROTATION_270, 180);
}

private final TextureView.SurfaceTextureListener surfaceTextureListener =
   new TextureView.SurfaceTextureListener() {
    @Override
    public void onSurfaceTextureAvailable(
       final SurfaceTexture texture, final int width, final int height) {
     openCamera(width, height);
    }

    @Override
    public void onSurfaceTextureSizeChanged(
       final SurfaceTexture texture, final int width, final int height) {
     configureTransform(width, height);
    }

    @Override
    public boolean onSurfaceTextureDestroyed(final SurfaceTexture texture) {
     return true;
    }

    @Override
    public void onSurfaceTextureUpdated(final SurfaceTexture texture) {}
   };

/**
 * Callback for Activities to use to initialize their data once the
 * selected preview size is known.
 */
public interface ConnectionCallback {
 void onPreviewSizeChosen(Size size, int cameraRotation);
```

```
}

private String cameraId;
private AutoFitTextureView textureView;
private Size previewSize;
private final CameraDevice.StateCallback stateCallback =
   new CameraDevice.StateCallback() {
    @Override
    public void onOpened(final CameraDevice cd) {
     cameraOpenCloseLock.release();
     cameraDevice = cd;
     createCameraPreviewSession();
    }

    @Override
    public void onDisconnected(final CameraDevice cd) {
     cameraOpenCloseLock.release();
     cd.close();
     cameraDevice = null;
    }

    @Override
    public void onError(final CameraDevice cd, final int error) {
     cameraOpenCloseLock.release();
     cd.close();
     cameraDevice = null;
     final Activity activity = getActivity();
     if (null != activity) {
       activity.finish();
     }
    }
   };

private final ConnectionCallback cameraConnectionCallback;
```

```
private CameraConnectionFragment(
    final ConnectionCallback connectionCallback,
    final OnImageAvailableListener imageListener,
    final int layout, final int inputSize) {
  this.cameraConnectionCallback = connectionCallback;
  this.imageListener = imageListener;
  this.layout = layout;
  this.inputSize = inputSize;
}

private void showToast(final String text) {
  final Activity activity = getActivity();
  if (activity != null) {
    activity.runOnUiThread(
        new Runnable() {
          @Override
          public void run() {
            Toast.makeText(activity, text, Toast.LENGTH_SHORT).show();
          }
        });
  }
}
private static Size chooseOptimalSize(
    final Size[] choices, final int width, final int height, final Size aspectRatio) {
  // Collect the supported resolutions that are at least as big as the preview Surface
  final List<Size> bigEnough = new ArrayList<Size>();

  final int minWidth = Math.max(width, MINIMUM_PREVIEW_SIZE);
  final int minHeight = Math.max(height, MINIMUM_PREVIEW_SIZE);

  for (final Size option : choices) {
    if (option.getHeight() >= minHeight && option.getWidth() >= minWidth) {
      LOGGER.i("Adding size: " + option.getWidth() + "x" + option.getHeight());
      bigEnough.add(option);
    } else {
```

```
      LOGGER.i("Not adding size: " + option.getWidth() + "x" + option.getHeight());
    }
  }

  // Pick the smallest of those, assuming we found any
  if (bigEnough.size() > 0) {
    final Size chosenSize = Collections.min(bigEnough, new CompareSizesByArea());
    LOGGER.i("Chosen size: " + chosenSize.getWidth() + "x" + chosenSize.getHeight());
    return chosenSize;
  } else {
    LOGGER.e("Couldn't find any suitable preview size");
    return choices[0];
  }
}

public static CameraConnectionFragment newInstance(
    final ConnectionCallback callback,
    final OnImageAvailableListener imageListener, final int layout, final int inputSize) {
  return new CameraConnectionFragment(callback, imageListener, layout, inputSize);
}

@Override
public View onCreateView(
    final LayoutInflater inflater, final ViewGroup container, final Bundle savedInstanceState) {
  return inflater.inflate(layout, container, false);
}

@Override
public void onViewCreated(final View view, final Bundle savedInstanceState) {
  textureView = (AutoFitTextureView) view.findViewById(R.id.texture);
}

@Override
public void onActivityCreated(final Bundle savedInstanceState) {
  super.onActivityCreated(savedInstanceState);
```

```
    }

    @Override
    public void onResume() {
      super.onResume();
      startBackgroundThread();

      // When the screen is turned off and turned back on, the SurfaceTexture is already
      // available, and "onSurfaceTextureAvailable" will not be called. In that case, we can open
      // a camera and start preview from here (otherwise, we wait until the surface is ready in
      // the SurfaceTextureListener).
      if (textureView.isAvailable()) {
        openCamera(textureView.getWidth(), textureView.getHeight());
      } else {
        textureView.setSurfaceTextureListener(surfaceTextureListener);
      }
    }

    @Override
    public void onPause() {
      closeCamera();
      stopBackgroundThread();
      super.onPause();
    }

    /**
     * Sets up member variables related to camera.
     *
     * @param width  The width of available size for camera preview
     * @param height  The height of available size for camera preview
     */
    private void setUpCameraOutputs(final int width, final int height) {
      final Activity activity = getActivity();
      final CameraManager manager = (CameraManager)
activity.getSystemService(Context.CAMERA_SERVICE);
```

```
        try {
          for (final String cameraId : manager.getCameraIdList()) {
            final CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);

            // We don't use a front facing camera in this sample.
            final Integer facing = characteristics.get(CameraCharacteristics.LENS_FACING);
            if (facing != null && facing == CameraCharacteristics.LENS_FACING_FRONT) {
              continue;
            }

            final StreamConfigurationMap map =

characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);

            if (map == null) {
              continue;
            }

            // For still image captures, we use the largest available size.
            final Size largest =
                Collections.max(
                    Arrays.asList(map.getOutputSizes(ImageFormat.YUV_420_888)),
                    new CompareSizesByArea());

            sensorOrientation = characteristics.get(CameraCharacteristics.SENSOR_ORIENTATION);

            // Danger, W.R.! Attempting to use too large a preview size could exceed the camera
            // bus' bandwidth limitation, resulting in gorgeous previews but the storage of
            // garbage capture data.
            previewSize =
                chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
                    inputSize, inputSize, largest);

            // We fit the aspect ratio of TextureView to the size of preview we picked.
            final int orientation = getResources().getConfiguration().orientation;
```

```
            if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
              textureView.setAspectRatio(previewSize.getWidth(), previewSize.getHeight());
            } else {
              textureView.setAspectRatio(previewSize.getHeight(), previewSize.getWidth());
            }

            CameraConnectionFragment.this.cameraId = cameraId;

            cameraConnectionCallback.onPreviewSizeChosen(previewSize, sensorOrientation);
            return;
          }
        } catch (final CameraAccessException e) {
          LOGGER.e(e, "Exception!");
        } catch (final NullPointerException e) {
          // Currently an NPE is thrown when the Camera2API is used but not supported on the
          // device this code runs.
          ErrorDialog.newInstance(getString(R.string.camera_error))
              .show(getChildFragmentManager(), FRAGMENT_DIALOG);
        }
      }


      private void openCamera(final int width, final int height) {
        setUpCameraOutputs(width, height);
        configureTransform(width, height);
        final Activity activity = getActivity();
        final CameraManager manager = (CameraManager)
activity.getSystemService(Context.CAMERA_SERVICE);
        try {
          if (!cameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
            throw new RuntimeException("Time out waiting to lock camera opening.");
          }
          manager.openCamera(cameraId, stateCallback, backgroundHandler);
        } catch (final CameraAccessException e) {
          LOGGER.e(e, "Exception!");
```

```
  } catch (final InterruptedException e) {
    throw new RuntimeException("Interrupted while trying to lock camera opening.", e);
  }
}


private void closeCamera() {
  try {
    cameraOpenCloseLock.acquire();
    if (null != captureSession) {
      captureSession.close();
      captureSession = null;
    }
    if (null != cameraDevice) {
      cameraDevice.close();
      cameraDevice = null;
    }
    if (null != previewReader) {
      previewReader.close();
      previewReader = null;
    }
  } catch (final InterruptedException e) {
    throw new RuntimeException("Interrupted while trying to lock camera closing.", e);
  } finally {
    cameraOpenCloseLock.release();
  }
}


private void startBackgroundThread() {
  backgroundThread = new HandlerThread("ImageListener");
  backgroundThread.start();
  backgroundHandler = new Handler(backgroundThread.getLooper());
}
```

```
/**
 * Stops the background thread and its {@link Handler}.
 */
private void stopBackgroundThread() {
 backgroundThread.quitSafely();
 try {
  backgroundThread.join();
  backgroundThread = null;
  backgroundHandler = null;
 } catch (final InterruptedException e) {
  LOGGER.e(e, "Exception!");
 }
}

private final CameraCaptureSession.CaptureCallback captureCallback =
  new CameraCaptureSession.CaptureCallback() {
   @Override
   public void onCaptureProgressed(
     final CameraCaptureSession session,
     final CaptureRequest request,
     final CaptureResult partialResult) {}

   @Override
   public void onCaptureCompleted(
     final CameraCaptureSession session,
     final CaptureRequest request,
     final TotalCaptureResult result) {}
  };


private void createCameraPreviewSession() {
 try {
  final SurfaceTexture texture = textureView.getSurfaceTexture();
  assert texture != null;
  texture.setDefaultBufferSize(previewSize.getWidth(), previewSize.getHeight());
```

```
        final Surface surface = new Surface(texture);
        previewRequestBuilder =
cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
        previewRequestBuilder.addTarget(surface);

        LOGGER.i("Opening camera preview: " + previewSize.getWidth() + "x" +
previewSize.getHeight());


        previewReader =
            ImageReader.newInstance(
                previewSize.getWidth(), previewSize.getHeight(), ImageFormat.YUV_420_888, 2);

        previewReader.setOnImageAvailableListener(imageListener, backgroundHandler);
        previewRequestBuilder.addTarget(previewReader.getSurface());

        // Here, we create a CameraCaptureSession for camera preview.
        cameraDevice.createCaptureSession(
            Arrays.asList(surface, previewReader.getSurface()),
            new CameraCaptureSession.StateCallback() {

              @Override
              public void onConfigured(final CameraCaptureSession cameraCaptureSession) {
                // The camera is already closed
                if (null == cameraDevice) {
                  return;
                }

                // When the session is ready, we start displaying the preview.
                captureSession = cameraCaptureSession;
                try {
                  // Auto focus should be continuous for camera preview.
                  previewRequestBuilder.set(
```

```
                    CaptureRequest.CONTROL_AF_MODE,
                    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
                // Flash is automatically enabled when necessary.
                previewRequestBuilder.set(
                    CaptureRequest.CONTROL_AE_MODE,
CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);

                // Finally, we start displaying the camera preview.
                previewRequest = previewRequestBuilder.build();
                captureSession.setRepeatingRequest(
                    previewRequest, captureCallback, backgroundHandler);
              } catch (final CameraAccessException e) {
                LOGGER.e(e, "Exception!");
              }
            }

            @Override
            public void onConfigureFailed(final CameraCaptureSession cameraCaptureSession) {
              showToast("Failed");
            }
          },
          null);
      } catch (final CameraAccessException e) {
        LOGGER.e(e, "Exception!");
      }
    }


    private void configureTransform(final int viewWidth, final int viewHeight) {
      final Activity activity = getActivity();
      if (null == textureView || null == previewSize || null == activity) {
        return;
      }
      final int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
      final Matrix matrix = new Matrix();
```

```
  final RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
  final RectF bufferRect = new RectF(0, 0, previewSize.getHeight(), previewSize.getWidth());
  final float centerX = viewRect.centerX();
  final float centerY = viewRect.centerY();
  if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
    bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());
    matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
    final float scale =
        Math.max(
            (float) viewHeight / previewSize.getHeight(),
            (float) viewWidth / previewSize.getWidth());
    matrix.postScale(scale, scale, centerX, centerY);
    matrix.postRotate(90 * (rotation - 2), centerX, centerY);
  } else if (Surface.ROTATION_180 == rotation) {
    matrix.postRotate(180, centerX, centerY);
  }
  textureView.setTransform(matrix);
}


static class CompareSizesByArea implements Comparator<Size> {
  @Override
  public int compare(final Size lhs, final Size rhs) {
    // We cast here to ensure the multiplications won't overflow
    return Long.signum(
        (long) lhs.getWidth() * lhs.getHeight() - (long) rhs.getWidth() * rhs.getHeight());
  }
}

public class DetectorActivity extends CameraActivity implements OnImageAvailableListener {
private static final Logger LOGGER = new Logger();

// Configuration values for the prepackaged multibox model.
private static final int MB_NUM_LOCATIONS = 784;
private static final int MB_INPUT_SIZE = 224;
```

```
    private static final int MB_IMAGE_MEAN = 128;
    private static final float MB_IMAGE_STD = 128;
    private static final String MB_INPUT_NAME = "ResizeBilinear";
    private static final String MB_OUTPUT_NAMES =
"output_locations/Reshape,output_scores/Reshape";
    private static final String MB_MODEL_FILE = "file:///android_asset/multibox_model.pb";
    private static final String MB_LOCATION_FILE =
        "file:///android_asset/multibox_location_priors.txt";

    // Configuration values for tiny-yolo-voc. Note that the graph is not included with TensorFlow
and
    // must be manually placed in the assets/ directory by the user.
    // Graphs and models downloaded from http://pjreddie.com/darknet/yolo/ may be converted e.g.
via
    // DarkFlow (https://github.com/thtrieu/darkflow). Sample command:
    // ./flow --model cfg/tiny-yolo-voc.cfg --load bin/tiny-yolo-voc.weights --savepb --
verbalise=True
    private static final String YOLO_MODEL_FILE = "file:///android_asset/graph-tiny-yolo-
voc.pb";
    private static final int YOLO_INPUT_SIZE = 416;
    private static final String YOLO_INPUT_NAME = "input";
    private static final String YOLO_OUTPUT_NAMES = "output";
    private static final int YOLO_BLOCK_SIZE = 32;

    // Default to the included multibox model.
    private static final boolean USE_YOLO = false;

    private static final int CROP_SIZE = USE_YOLO ? YOLO_INPUT_SIZE : MB_INPUT_SIZE;

    // Minimum detection confidence to track a detection.
    private static final float MINIMUM_CONFIDENCE = USE_YOLO ? 0.0f : 0.1f;

    private static final boolean MAINTAIN_ASPECT = USE_YOLO;

    private static final boolean SAVE_PREVIEW_BITMAP = false;
```

```
private static final float TEXT_SIZE_DIP = 10;

private Integer sensorOrientation;

private Classifier detector;

private int previewWidth = 0;
private int previewHeight = 0;
private byte[][] yuvBytes;
private int[] rgbBytes = null;
private Bitmap rgbFrameBitmap = null;
private Bitmap croppedBitmap = null;

private boolean computing = false;

private long timestamp = 0;

private Matrix frameToCropTransform;
private Matrix cropToFrameTransform;

private Bitmap cropCopyBitmap;

private MultiBoxTracker tracker;

private byte[] luminance;

private BorderedText borderedText;

private long lastProcessingTimeMs;

@Override
public void onPreviewSizeChosen(final Size size, final int rotation) {
 final float textSizePx =
    TypedValue.applyDimension(
```

```
            TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP,
getResources().getDisplayMetrics());
        borderedText = new BorderedText(textSizePx);
        borderedText.setTypeface(Typeface.MONOSPACE);

        tracker = new MultiBoxTracker(getResources().getDisplayMetrics());

        try {
         if (USE_YOLO) {
          detector =
              TensorFlowYoloDetector.create(
                  getAssets(),
                  YOLO_MODEL_FILE,
                  YOLO_INPUT_SIZE,
                  YOLO_INPUT_NAME,
                  YOLO_OUTPUT_NAMES,
                  YOLO_BLOCK_SIZE);
         } else {
          detector =
              TensorFlowMultiBoxDetector.create(
                  getAssets(),
                  MB_MODEL_FILE,
                  MB_LOCATION_FILE,
                  MB_NUM_LOCATIONS,
                  MB_INPUT_SIZE,
                  MB_IMAGE_MEAN,
                  MB_IMAGE_STD,
                  MB_INPUT_NAME,
                  MB_OUTPUT_NAMES);
         }
        } catch (final Exception e) {
         throw new RuntimeException("Error initializing TensorFlow!", e);
        }

        previewWidth = size.getWidth();
```

```
previewHeight = size.getHeight();

final Display display = getWindowManager().getDefaultDisplay();
final int screenOrientation = display.getRotation();

LOGGER.i("Sensor orientation: %d, Screen orientation: %d", rotation, screenOrientation);

sensorOrientation = rotation + screenOrientation;

LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
rgbBytes = new int[previewWidth * previewHeight];
rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight, Config.ARGB_8888);
croppedBitmap = Bitmap.createBitmap(CROP_SIZE, CROP_SIZE, Config.ARGB_8888);

frameToCropTransform =
    ImageUtils.getTransformationMatrix(
        previewWidth, previewHeight,
        CROP_SIZE, CROP_SIZE,
        sensorOrientation, MAINTAIN_ASPECT);

cropToFrameTransform = new Matrix();
frameToCropTransform.invert(cropToFrameTransform);
yuvBytes = new byte[3][];

trackingOverlay = (OverlayView) findViewById(R.id.tracking_overlay);
trackingOverlay.addCallback(
    new DrawCallback() {
      @Override
      public void drawCallback(final Canvas canvas) {
        tracker.draw(canvas);
        if (isDebug()) {
          tracker.drawDebug(canvas);
        }
      }
    });
```

```
addCallback(
    new DrawCallback() {
      @Override
      public void drawCallback(final Canvas canvas) {
        if (!isDebug()) {
          return;
        }
        final Bitmap copy = cropCopyBitmap;
        if (copy == null) {
          return;
        }

        final int backgroundColor = Color.argb(100, 0, 0, 0);
        canvas.drawColor(backgroundColor);

        final Matrix matrix = new Matrix();
        final float scaleFactor = 2;
        matrix.postScale(scaleFactor, scaleFactor);
        matrix.postTranslate(
            canvas.getWidth() - copy.getWidth() * scaleFactor,
            canvas.getHeight() - copy.getHeight() * scaleFactor);
        canvas.drawBitmap(copy, matrix, new Paint());

        final Vector<String> lines = new Vector<String>();
        if (detector != null) {
          final String statString = detector.getStatString();
          final String[] statLines = statString.split("\n");
          for (final String line : statLines) {
            lines.add(line);
          }
        }
        lines.add("");

        lines.add("Frame: " + previewWidth + "x" + previewHeight);
```

```
        lines.add("Crop: " + copy.getWidth() + "x" + copy.getHeight());
        lines.add("View: " + canvas.getWidth() + "x" + canvas.getHeight());
        lines.add("Rotation: " + sensorOrientation);
        lines.add("Inference time: " + lastProcessingTimeMs + "ms");

        borderedText.drawLines(canvas, 10, canvas.getHeight() - 10, lines);
      }
    });
}

OverlayView trackingOverlay;

@Override
public void onImageAvailable(final ImageReader reader) {
 Image image = null;

 ++timestamp;
 final long currTimestamp = timestamp;

 try {
  image = reader.acquireLatestImage();

  if (image == null) {
   return;
  }

  Trace.beginSection("imageAvailable");

  final Plane[] planes = image.getPlanes();
  fillBytes(planes, yuvBytes);

  tracker.onFrame(
      previewWidth,
      previewHeight,
      planes[0].getRowStride(),
```

```
        sensorOrientation,
        yuvBytes[0],
        timestamp);
    trackingOverlay.postInvalidate();

    // No mutex needed as this method is not reentrant.
    if (computing) {
      image.close();
      return;
    }
    computing = true;

    final int yRowStride = planes[0].getRowStride();
    final int uvRowStride = planes[1].getRowStride();
    final int uvPixelStride = planes[1].getPixelStride();
    ImageUtils.convertYUV420ToARGB8888(
        yuvBytes[0],
        yuvBytes[1],
        yuvBytes[2],
        rgbBytes,
        previewWidth,
        previewHeight,
        yRowStride,
        uvRowStride,
        uvPixelStride,
        false);

    image.close();
  } catch (final Exception e) {
    if (image != null) {
      image.close();
    }
    LOGGER.e(e, "Exception!");
    Trace.endSection();
    return;
```

```
}

rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);
final Canvas canvas = new Canvas(croppedBitmap);
canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);

// For examining the actual TF input.
if (SAVE_PREVIEW_BITMAP) {
 ImageUtils.saveBitmap(croppedBitmap);
}

if (luminance == null) {
 luminance = new byte[yuvBytes[0].length];
}
System.arraycopy(yuvBytes[0], 0, luminance, 0, luminance.length);

runInBackground(
   new Runnable() {
     @Override
     public void run() {
       final long startTime = SystemClock.uptimeMillis();
       final List<Classifier.Recognition> results = detector.recognizeImage(croppedBitmap);
       lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

       cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
       final Canvas canvas = new Canvas(cropCopyBitmap);
       final Paint paint = new Paint();
       paint.setColor(Color.RED);
       paint.setStyle(Style.STROKE);
       paint.setStrokeWidth(2.0f);

       final List<Classifier.Recognition> mappedRecognitions =
          new LinkedList<Classifier.Recognition>();

       for (final Classifier.Recognition result : results) {
```

```
          final RectF location = result.getLocation();
          if (location != null && result.getConfidence() >= MINIMUM_CONFIDENCE) {
            canvas.drawRect(location, paint);

            cropToFrameTransform.mapRect(location);
            result.setLocation(location);
            mappedRecognitions.add(result);
          }
        }

        tracker.trackResults(mappedRecognitions, luminance, currTimestamp);
        trackingOverlay.postInvalidate();

        requestRender();
        computing = false;
      }
    });

  Trace.endSection();
}

@Override
protected int getLayoutId() {
  return R.layout.camera_connection_fragment_tracking;
}

@Override
protected int getDesiredPreviewFrameSize() {
  return CROP_SIZE;
}

@Override
public void onSetDebug(final boolean debug) {
  detector.enableStatLogging(debug);
}
```

```
}
  public static class ErrorDialog extends DialogFragment {
    private static final String ARG_MESSAGE = "message";

    public static ErrorDialog newInstance(final String message) {
      final ErrorDialog dialog = new ErrorDialog();
      final Bundle args = new Bundle();
      args.putString(ARG_MESSAGE, message);
      dialog.setArguments(args);
      return dialog;
    }

public class TensorFlowImageClassifier implements Classifier {
  static {
    System.loadLibrary("tensorflow_demo");
  }

  private static final String TAG = 'TensorFlowImageClassifier";

  // Only return this many results with at least this confidence.
  private static final int MAX_RESULTS = 3;
  private static final float THRESHOLD = 0.1f;

  // Config values.
  private String inputName;
  private String outputName;
  private int inputSize;
  private int imageMean;
  private float imageStd;

  // Pre-allocated buffers.
  private Vector<String> labels = new Vector<String>();
  private int[] intValues;
  private float[] floatValues;
  private float[] outputs;
```

```
private String[] outputNames;

private TensorFlowInferenceInterface inferenceInterface;

private TensorFlowImageClassifier() {}

/**
 * Initializes a native TensorFlow session for classifying images.
 *
 * @param assetManager The asset manager to be used to load assets.
 * @param modelFilename The filepath of the model GraphDef protocol buffer.
 * @param labelFilename The filepath of label file for classes.
 * @param numClasses The number of classes output by the model.
 * @param inputSize The input size. A square image of inputSize x inputSize is assumed.
 * @param imageMean The assumed mean of the image values.
 * @param imageStd The assumed std of the image values.
 * @param inputName The label of the image input node.
 * @param outputName The label of the output node.
 * @throws IOException
 */
public static Classifier create(
    AssetManager assetManager,
    String modelFilename,
    String labelFilename,
    int numClasses,
    int inputSize,
    int imageMean,
    float imageStd,
    String inputName,
    String outputName)
    throws IOException {
  TensorFlowImageClassifier c = new TensorFlowImageClassifier();
  c.inputName = inputName;
  c.outputName = outputName;
```

```
  // Read the label names into memory.
  // TODO(andrewharp): make this handle non-assets.
  String actualFilename = labelFilename.split("file:///android_asset/")[1];
  Log.i(TAG, "Reading labels from: " + actualFilename);
  BufferedReader br = null;
  br = new BufferedReader(new InputStreamReader(assetManager.open(actualFilename)));
  String line;
  while ((line = br.readLine()) != null) {
    c.labels.add(line);
  }
  br.close();
  Log.i(TAG, "Read " + c.labels.size() + ", " + numClasses + " specified");

  c.inputSize = inputSize;
  c.imageMean = imageMean;
  c.imageStd = imageStd;

  // Pre-allocate buffers.
  c.outputNames = new String[] {outputName};
  c.intValues = new int[inputSize * inputSize];
  c.floatValues = new float[inputSize * inputSize * 3];
  c.outputs = new float[numClasses];

  c.inferenceInterface = new TensorFlowInferenceInterface();

  final int status = c.inferenceInterface.initializeTensorFlow(assetManager, modelFilename);
  if (status != 0) {
    Log.e(TAG, "TF init status: " + status);
    throw new RuntimeException("TF init status (" + status + ") != 0");
  }
  return c;
}
```

```
  @Override
  public void enableStatLogging(boolean debug) {
    inferenceInterface.enableStatLogging(debug);
  }

  @Override
  public String getStatString() {
    return inferenceInterface.getStatString();
  }

  @Override
  public void close() {
    inferenceInterface.close();
  }
}
    @Override
    public Dialog onCreateDialog(final Bundle savedInstanceState) {
      final Activity activity = getActivity();
      return new AlertDialog.Builder(activity)
          .setMessage(getArguments().getString(ARG_MESSAGE))
          .setPositiveButton(
              android.R.string.ok,
              new DialogInterface.OnClickListener() {
                @Override
                public void onClick(final DialogInterface dialogInterface, final int i) {
                  activity.finish();
                }
              })
          .create();
    }
  }
}
```

ДОДАТОК Б

Комп'ютерна презентація

Deep Neural Networks for Mobile Platforms

Дослідження та програмна реалізація
алгоритмів обробки зображень
на мобільних платформах

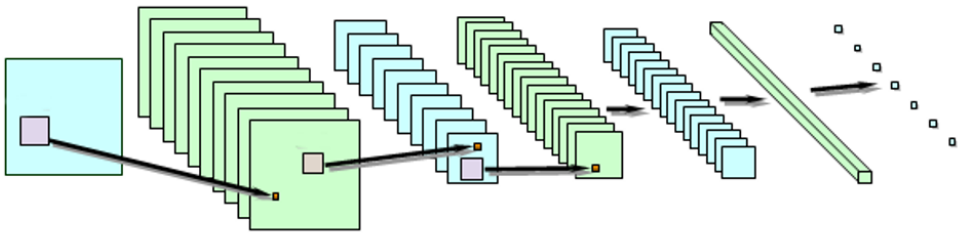
студент групи СП-16дм
Прядко К.М.

керівник проекту Щербакова М.Є.

2018

Рисунок Б.1 – Слайд №1

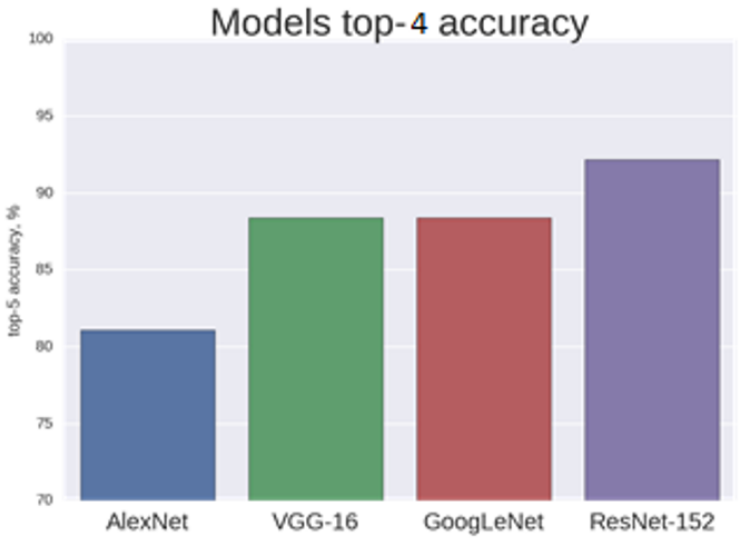
Нейронні мережі
Згорточні нейронні мережі



DNN for Mobile Platforms 2 / 17

Рисунок Б.2 – Слайд №2

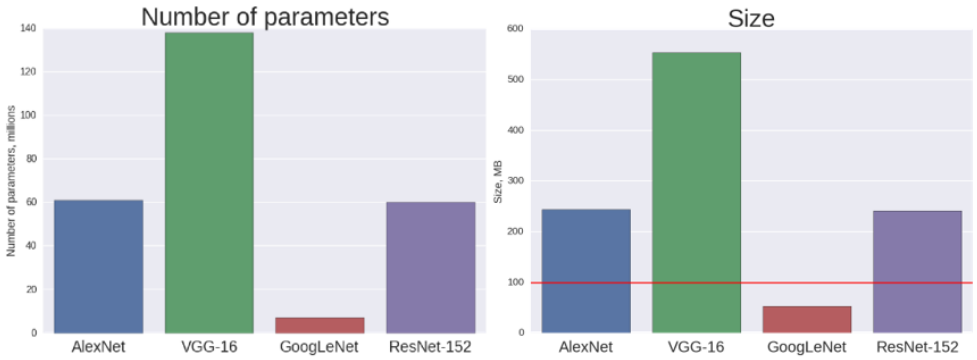
Проблеми етапу розгортання
Якість моделей



Alex Krizhevsky, et al. ImageNet Classification with Deep Convolutional Neural Networks, 2012
Christian Szegedy, et al. Going Deeper with Convolutions, 2014
K. Simonyan, et al. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014
Kaiming He, et al. Deep Residual Learning for Image Recognition, 2015

Рисунок Б.3 – Слайд №3

Проблеми етапу розгортання
Розмір моделей



Для завантаження програма з розміром більш ніж 100MB потребує підключення до WiFi

Рисунок Б.4 – Слайд №4

Проблеми етапу розгортання
Виконання моделі

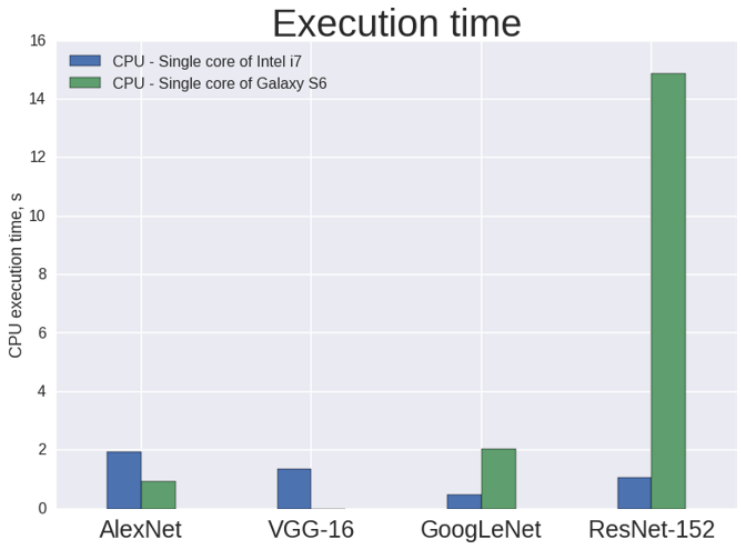
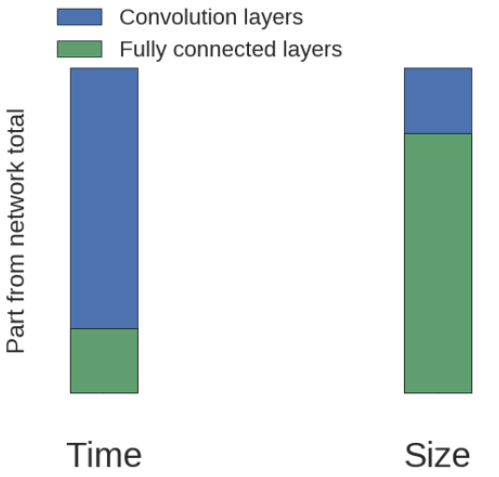


Рисунок Б.5 – Слайд №5

Огляд проблеми
Приблизний розподіл шарів



Багатошарова нейромережа більш ніж Згорточна нейромережа у розмірі.
Згорточна мережа потребує більше часу для розподілу шарів.
Обраному приладу необхідно зберігати в RAM карту ознак хоча б одного шару.

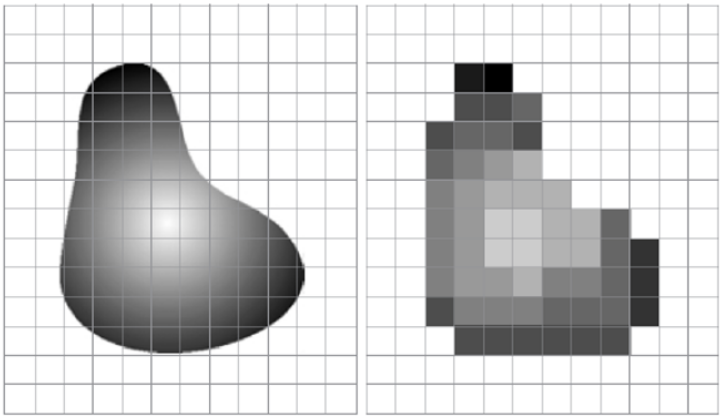
Рисунок Б.6 – Слайд №6

Огляд проблеми
Важливі параметри

Ширина та висота карти ознак впливає на час виконання.
Глибина карти ознак впливає на розмір моделі.

Рисунок Б.7 – Слайд №7

Квантизація

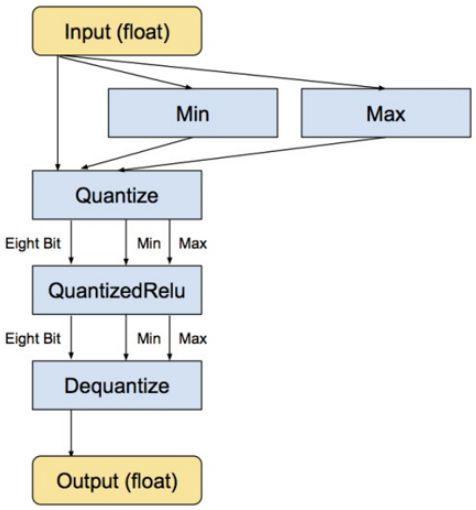


Pete Warden, How to Quantize Neural Networks with TensorFlow, 2016

Matthieu Courbariaux, et. al., BinaryConnect: Training Deep Neural Networks with binary weights during propagations, 2015

Рисунок Б.8 – Слайд №8

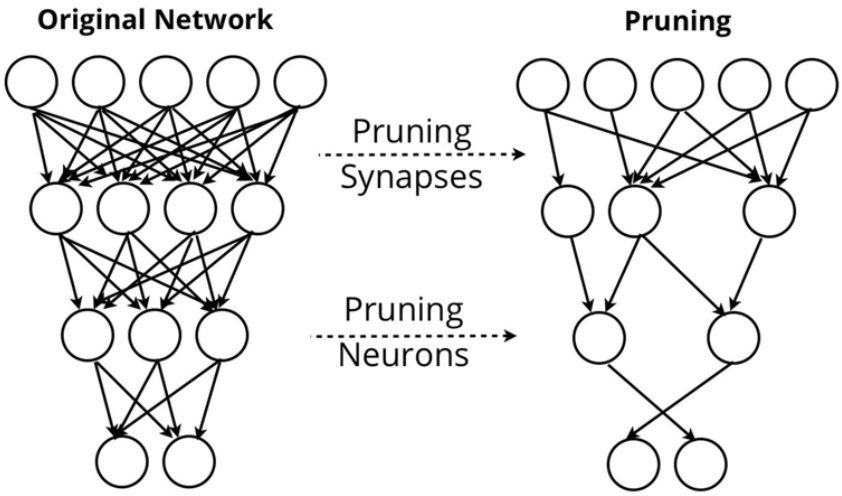
Схема квантизації



Кожна операція розраховує Min та Max значень, які використовуються для змінення маштабу
Значення Min та Max вибираються з дійсних чисел.

Рисунок Б.9 – Слайд №9

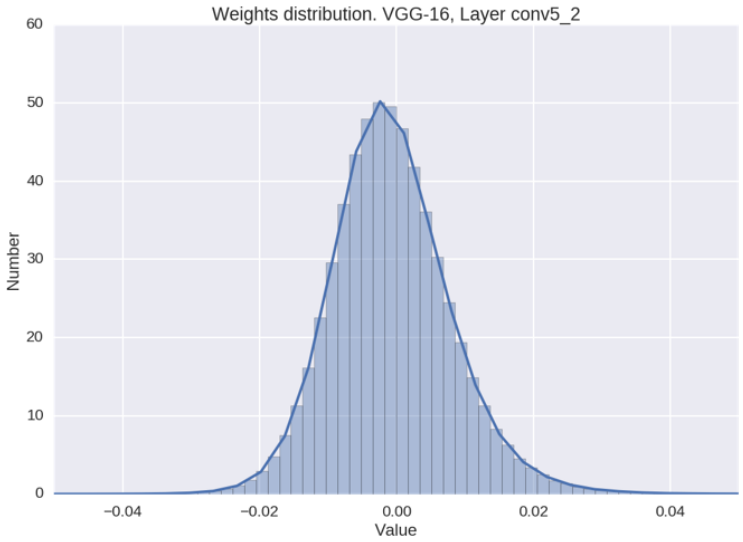
Спрощення



Ідеєю цього методу є обрізання неважливих зв'язків. Складність у тому, який конкретно зв'язок є «неважливим».

Рисунок Б.10 – Слайд №10

Спрощення
Неважливі критерії



Yann Le Cun, et al. Optimal Brain Damage, 1990
Babak Hassibi, et al. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon, 1992
Song Han, et al., Learning both Weights and Connections for Efficient Neural Networks, 2015

Рисунок Б.11 – Слайд №11

Спрощення
Алгоритм спрощення

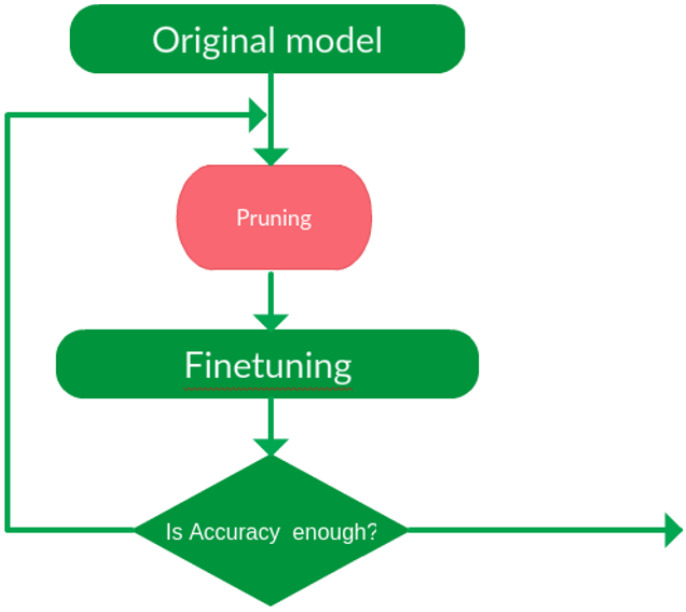
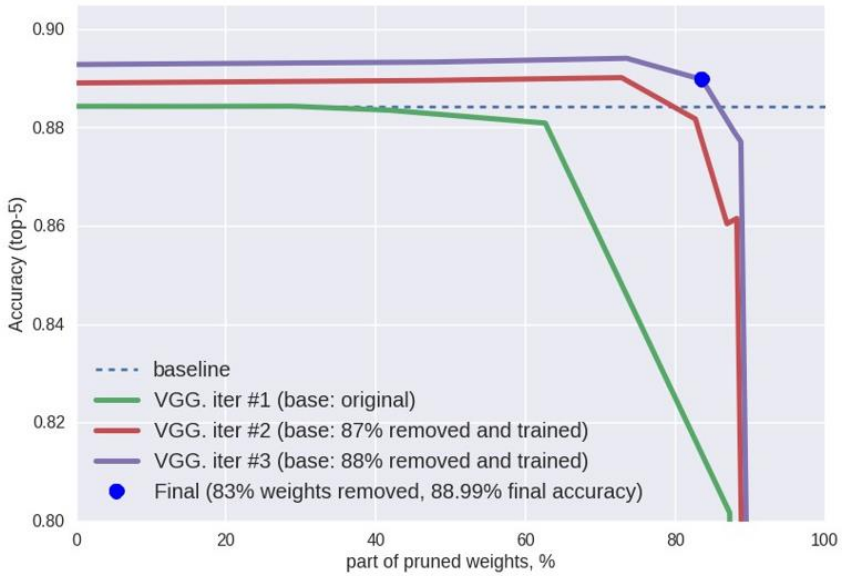


Рисунок Б.12 – Слайд №12

Приклад VGG



Song Han, DSD: Regularizing Deep Neural Networks with Dense-Sparse-Dense Training Flow, 2016

Рисунок Б.13 – Слайд №13

Спрощення VGG
Графік розподілу ваг після спрощення

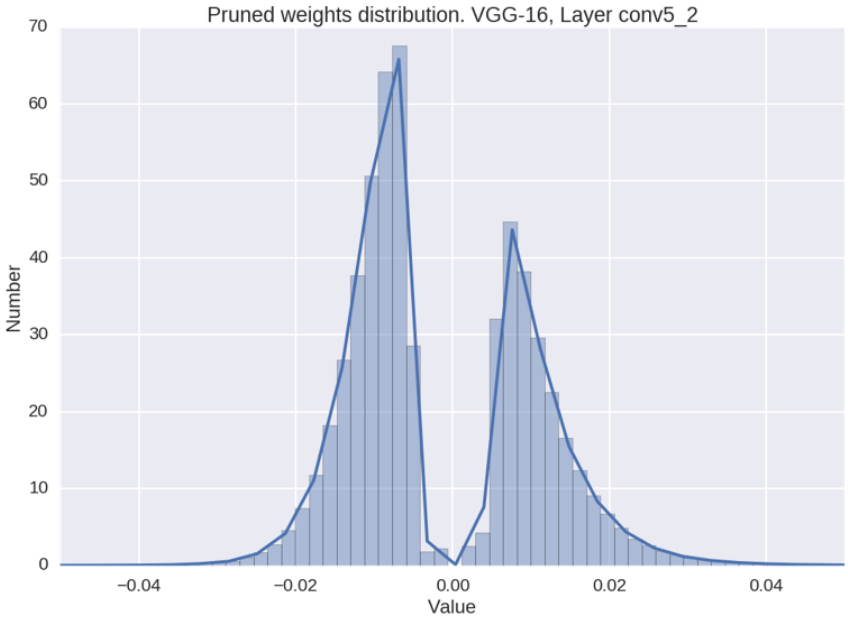


Рисунок Б.14 – Слайд №14

Спрощення ResNet-152

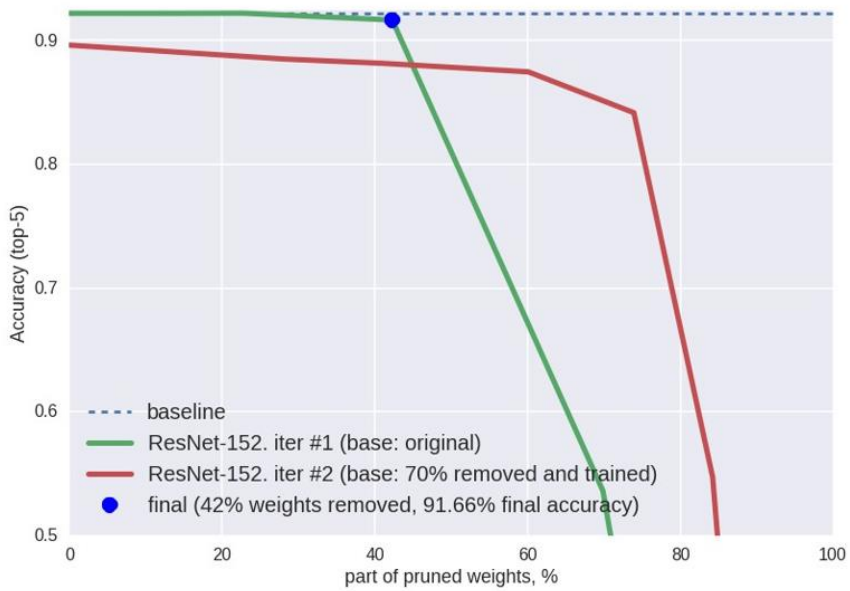
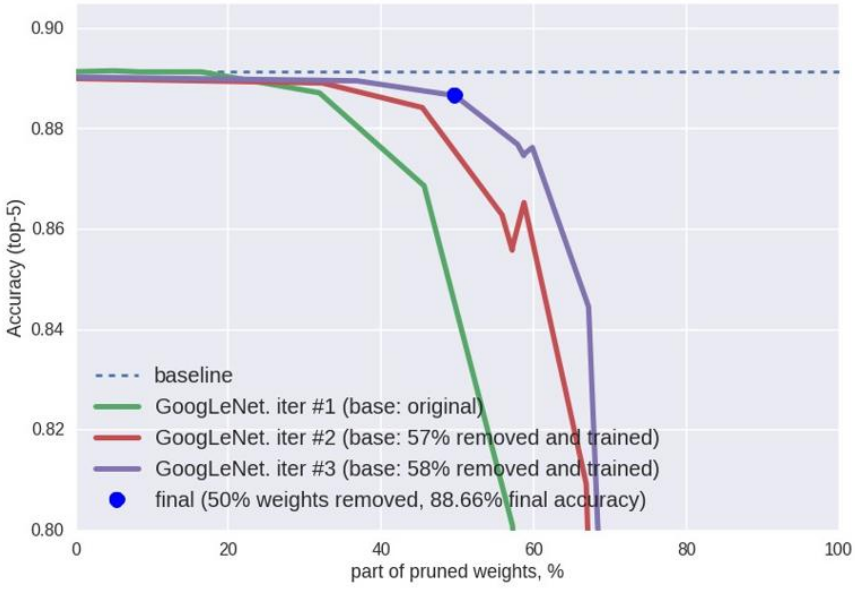


Рисунок Б.15 – Слайд №15

Спрощення GoogLeNet



Hao Li, et.al., Pruning Filters for Efficient ConvNets, 2016

Рисунок Б.16 – Слайд №16

ВИСНОВКИ

Глубокі згорточні нейромережі забезпечує чудову якість, але потребує дуже потужних обчислювальних приладів.
Існує кілька простих на корисних підходів для зменшення необхідного розміру пам'яті та часу виконання без збільшення витрат на апаратне забезпечення.

Рисунок Б.17 – Слайд №17