Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження згорткових нейронних мереж в задачі розпізнавання об'єктів на статичних зображеннях
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-1
Крят В.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Кулішова Н.Є
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Кряту Вадиму Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження згорткових нейронних мереж в задачі розпізнавання об'єктів на статичних зображеннях

затверджена наказом університету від 30 жовтня 2020 р. № 1497 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 грудня 2020 р.

3. Вихідні дані до роботи Науково-технічні публікації, Інтернет-ресурси за темою, дані статей, публікації в наукових журналах, кольорові цифрові зображення у банках даних із відкритим доступом

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Огляд існуючих нейронних мереж для вирішення задачі розпізнавання
3) Розробка системи та імітаційне моделювання

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Рисунок 1 – Залежність точності класифікації та коефіцієнту втрати помилки від кількості ітерацій

Рисунок 2 – Порівняння точності класифікації від вибору попередньої обробки зображень

Рисунок 3 – Архітектура Web-сервісу

Рисунок 4 – Приклад зображень з набору даних

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Кулішова Н.Є		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	01.11.2020	виконано
2	Аналіз предметної галузі та постановка задачі	02.11.2020 – 07.11.2020	виконано
3	Дослідження методів розпізнавання	07.11.2020 – 12.11.2020	виконано
4	Аналіз існуючих проблем даної предметної області	12.11.2020 – 17.11.2020	виконано
5	Розробка системи розпізнавання	17.11.2020 – 21.11.2020	виконано
6	Створення імітаційної моделі	21.11.2020 – 26.11.2020	виконано
7	Тестування і опрацювання імітаційної моделі	26.11.2020 – 28.11.2020	виконано
8	Оформлення пояснювальної записки	28.11.2020 – 29.11.2020	виконано
9	Оформлення графічних матеріалів	30.11.2020 – 01.12.2020	виконано
10	Попередній захист	14.12.2020	виконано
11	Захист перед ЕК	15.12.2020	

Дата видачі завдання 01 11 2020 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 86 с., 2 табл., 36 рис., 2 дод., 21 джерело.

ГЛИБОКЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КЛАСИФІКАЦІЯ, РОЗПІЗНАВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, AWS, GOOGLENET, KERAS, NODE.JS, PYTHON.

В атестаційній роботі розглядається задача розпізнавання моделі велосипеда за його зображенням.

Об’єктом дослідження є згорткова нейронна мережа прямого розповсюдження.

Предметом дослідження є процес розпізнавання зображень за допомогою згорткових нейронних мереж.

Метою атестаційної роботи є створення додатку для автоматичного розпізнавання моделі велосипеда за його зображенням на платформі Node.js з використанням хмарних технологій AWS, бібліотеки Keras та мови програмування Python.

Методами досліджень є аналіз існуючих методів розпізнавання, моделювання системи розпізнавання, вирішення практичних завдань та проведення порівняльного аналізу.

Результатом атестаційної роботи є розроблений додаток для автоматичного розпізнавання моделі велосипеда за його зображенням.

РЕФЕРАТ

Пояснительная записка: 86 с., 2 табл., 36 рис., 2 прил., 21 источник.

ГЛУБОКЕ ОБУЧЕНИЕ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, КЛАССИФИКАЦИЯ, РАСПОЗНАВАНИЕ, СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, AWS, GOOGLENET, KERAS, NODE.JS, PYTHON.

В аттестационной работе рассматривается задача распознавания модели велосипеда по его изображению.

Объектом исследования является свёрточная нейронная сеть прямого распространения.

Предметом исследования является процесс распознавания изображений с помощью свёрточных нейронных сетей.

Целью аттестационной работы является создание приложения для автоматического распознавания модели велосипеда по его изображению на платформе Node.js с использованием облачных технологий AWS, библиотеки Keras и языка программирования Python.

Методами исследований является анализ существующих методов распознавания, моделирования системы распознавания, решение практических задач и проведения сравнительного анализа.

Результатом аттестационной работы является разработано приложение для автоматического распознавания модели велосипеда по его изображению.

ABSTRACT

Explanatory note: 86 p., 2 tab., 36 fig., 2 app., 21 sources.

ARTIFICIAL INTELLIGENCE, AWS, CLASSIFICATION, CONVOLUTIONAL NEURAL NETWORK, DEEP LEARNING, GOOGLENET, KERAS, NODE.JS, PYTHON, RECOGNITION.

In the certification work, the task of recognizing a bicycle model by its image is considered.

The object of the research is a convolutional neural network of direct propagation.

The subject of the research is the process of image recognition using convolutional neural networks.

The aim of the certification work is to create an application for automatic visual recognition of a bicycle model from its image on the Node.js platform using AWS cloud technologies, the Keras library and the Python programming language.

Research methods include the analysis of existing recognition methods, modeling the recognition system, solving practical problems and conducting a comparative analysis.

The result of the certification work is implementation of an application for automatic recognition of a bicycle model by its image.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – convolutional neural network – згорткова нейронна мережа;

CPU – Central Processing Unit – центральний процесор;

DCNN – deep convolutional neural network – глибинна згорткова нейронна мережа;

GPU – Graphics Processing Unit – графічний процесор;

HOG – Histogram of Oriented Gradient – гістограма орієнтованих градієнтів;

HSV – колірна модель, заснована на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і значенні кольору (Value);

ImageNet – база анотованих зображень, призначених для обробки методами розпізнавання зображень та машинного зору;

ReLU – rectified linear unit – випрямлена лінійна одиниця;

RGB – red, green, blue – адитивна колірна модель.

ВСТУП

Методи машинного навчання є невід’ємною частиною сучасних підходів до розпізнавання об'єктів. Щоб підвищити їх продуктивність, ми можемо збирати великі набори даних, вивчати більш потужні моделі і використовувати більш досконалі методи для запобігання перенавчання. До недавнього часу набори даних помічених зображень були відносно невеликими – близько десятків тисяч зображень (наприклад, NORB, 19 Caltech-101 / 256,8, 10 і CIFAR-10/10014). Прості завдання розпізнавання можуть бути вирішені досить добре з наборами даних такого розміру, особливо якщо вони доповнені перетвореннями. Але об'єкти в реальному світі демонструють значну мінливість, тому, щоб навчитися їх розпізнавати, необхідно використовувати набагато більші навчальні набори. І дійсно, недоліки невеликих наборів даних зображень були широко визнані, але тільки недавно стало можливим збирати помічені набори даних з мільйонами зображень. Нові більші набори даних включають LabelMe, який складається з сотень тисяч повністю сегментованих зображень, і ImageNet 7, який складається з понад 15 мільйонів помічених зображень з високою роздільною здатністю в більше ніж 22 000 категорій.

Для вивчення тисячі об'єктів з мільйонів зображень, нам потрібна модель з великою навчальною потужністю. Однак величезна складність завдання розпізнавання об'єктів означає, що ця проблема не може бути визначена навіть за допомогою такого великого набору даних, як ImageNet, тому модель також має мати багато попередніх знань, щоб компенсувати всі дані, яких у нас немає. Також неможливо постійно поглиблювати та розширювати архітектуру нейронних мереж, адже це вимагає багато обчислювальної потужності і глибинні мережі важко тренувати через проблему зникаючого градієнту. Як результат, із поглибленням мережі її ефективність починає швидко погіршуватися.

За останні три роки, в основному завдяки прогресу в області глибинного навчання, а точніше згорткових нейронних мереж, якість розпізнавання зображень і виявлення об'єктів стрімко поліпшувалася. Гарна новина полягає в тому, що велика частина цього прогресу є не тільки результатом більш потужного устаткування, великих наборів даних і більших моделей, але головним чином наслідком нових ідей, алгоритмів і поліпшених мережевих архітектур. Наприклад, кращими учасниками конкурсу ILSVRC 2014 року не використовувалися ніякі нові джерела даних, крім набору класифікаційних даних того ж конкурсу для цілей розпізнавання. Наприклад, GoogLeNet на ILSVRC 2014 фактично використовувала на 12 параметрів менше, ніж у виграшній архітектурі Крижевського, отриманій два роки тому, але при цьому вона значно точніша. Найбільші успіхи у виявленні об'єктів були отримані не тільки від використання глибинних мереж або більших моделей, а завдяки синергії глибинних архітектур і класичного комп'ютерного зору, так як в алгоритмі R-CNN Гіршіка.

Ще одним важливим фактором є те, що з постійним розвитком мобільних та інтегрованих обчислювальних систем, використання їх потужності і пам'яті набуває все більшого значення. Примітно, що міркування, які призвели до проектування глибинної архітектури, включали цей фактор, а не просто зациклювалися на показниках точності. Для більшості експериментів моделі були спроектовані так, щоб обчислювальний бюджет становив 1:5 мільярдів множень-додавань під час виведення, так що вони не перетворилися на суто академічну цікавість, а могли бути використані в реальному світі. навіть для великих наборів даних за розумною ціною.

Метою даної роботи є розробка ефективної архітектури глибинної нейронної мережі для комп'ютерного зору.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття задачі розпізнавання зображення

Розпізнавання образів є одним з основних завдань у багатьох інтелектуальних системах. Розпізнавання образів може бути визначено, як класифікація даних на основі отриманих знань або статичної інформації, отриманої із прикладів. Тобто це процес віднесення об'єкта за фіксованою групою його властивостей до одного об'єкту з множини образів за заздалегідь обумовленим правилом [1].

З визначення теорії розпізнавання образів можна виділи такі основні поняття, як образ, множина, вирішальне правило. Образ – це деякий структурований наближений опис досліджуваного об’єкта, явища чи процесу. Усі образи мають характерну рису, яка виявляється в тому, що ознайомлення з кінцевою кількістю явищ, які відносяться до однієї множини, надає можливість виділяти інших представників цієї множини. Множиною називається набір неповторюваних однотипних елементів, тобто кожен елемент може або належати множині або бути представником іншої.

Виділяють два типи множин:

– універсальна множина, тобто та множина яка включає всі можливі елементи;

– пуста множина – множина яка не містить жодного елемента;

Методика віднесення елемента до якогось образу називається вирішальним правилом. Невід’ємним поняттям розпізнавання образів є метрика. Вона визначає відстань між елементами множини, який дозволяє визначити міру схожості елементів між собою. Від вибору представлення образів і реалізації метрики залежить ефективність роботи програми: отой самий алгоритм розпізнавання з різними метриками буде помилятися з різною частотою.

Розпізнавання образів можна поділити на декілька підзадач:

– отримання вхідних даних за допомогою сенсорів, камер або наборів даних;

– формування векторів ознак шляхом вибору найбільш значущих ознак, за допомогою яких можливо виявити неперетинні множини класів. Основною і важливою задачею у теорії розпізнавання образів є вибір мінімальної кількості ознак, які описують образ найбільш інформативно у даній системі розпізнавання;

– класифікація на основі отриманих даних.

Системи розпізнавання образів – це засоби, побудовані на основі систематичних теоретичних та експериментальних досліджень, які здатні проводити співвіднесення описів отриманих з об’єктів, явищ, процесів до відповідних класів. Підходи до комп'ютеризації розпізнавання образів можна розділити на дві основні категорії: методи, засновані на теорії рішень та структурний аналіз. Перша категорія має справу з образами, які описуються кількісними дескрипторами, а друга категорія методів орієнтована на образи, які можна добре описати символьною інформацією та властивостями і взаємовідносинами між цими символами [2].

Розробка системи розпізнавання образів відбувається кількома етапами:

– розробка тренувального набору. Тренувальна вибірка – це колекція об'єктів для яких заздалегідь відомі їхні образи;

– вибір моделі представлення об'єктів. Найбільш зручним математичним описом вважається векторний опис образів, де кожному образу ставиться у відповідність деякий вектор ознак;

– виділення ознак. Специфіка опису об'єктів ознаками розпізнавання простежується в їх класифікації. Ознаки розпізнавання можна поділити на три категорії: за способом опису об'єктів (кількісні, якісні і логічні), за характером інформації (імовірнісні, детерміновані), за джерелом отриманої інформації (фізичні, математичні та структурні ознаки);

– розробка класифікаційного правила, тобто визначення методу співвіднесення вектору ознак образів деякому класу;

– навчання алгоритму. Навчанням зазвичай називають процес вироблення в деякій системі тієї чи іншої реакції на фактори зовнішніх схожих сигналів шляхом їх багаторазового впливу на систему;

– оцінка достовірності класифікації образів. Ця оцінка необхідна, щоб особа, яка приймає рішення (це може бути і технічна система), пов'язане з віднесенням образу того чи іншого класу, могла оцінити величину втрат, пов'язаних з неправильною класифікацією.

Схема роботи системи розпізнавання представлена на рисунку 1.1:

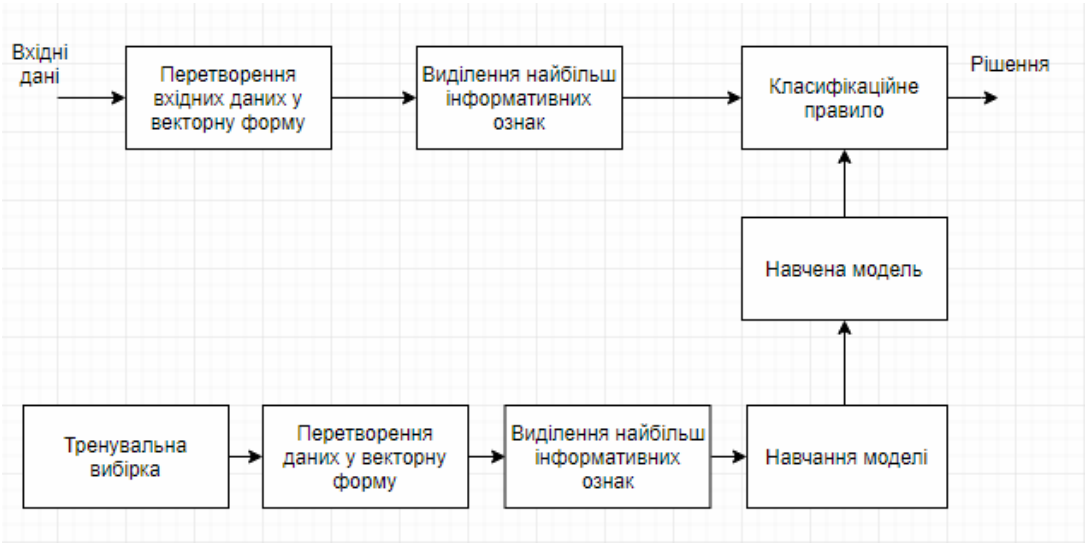

Рисунок 1.1 – Схема роботи системи розпізнавання

1.2 Області застосування технології розпізнавання

Області застосування і дослідження методів глибинного навчання значно розвинулися за невеликий проміжок часу. Класифікація зображень була першим додатком, в якому глибинні нейронні мережі почали показувати неймовірні результати. Змагання ImageNet підштовхнуло безліч груп до досліджень в цій області, з неймовірними результатами в

класифікації зображень, локалізації одного і декількох об'єктів і виявлення об'єктів в відео.

Нижче представлені області застосування технології розпізнавання зображень:

– автомобільна промисловість. Над безпілотними автомобілями працюють не тільки традиційні виробники автомобілів, але і технологічні гіганти беруть в свої руки виробництво таких автомобілів. Існує багато причин, які спонукають розвиток в даному напрямі, а саме: зниження кількості дорожньо-транспортних пригод, дотримання правил дорожнього руху тощо. Наприклад, на виставці CES Cisco оголосила про партнерство з традиційною компанією-виробником автомобілів Hyundai задля допомоги в забезпеченні автономних автомобілів;

– індустрія охорони здоров'я. Технологія розпізнавання зображень надає величезну допомогу в галузі охорони здоров'я. У мікрохірургічних процедурах в сфері охорони здоров'я за допомогою роботів використовуються методи комп'ютерного зору і розпізнавання зображень. Використання цього методу почастішало за останнє десятиліття завдяки досягненням в області машинного навчання і штучного інтелекту. Виявлення емоцій в режимі реального часу також можна використовувати для виявлення емоцій пацієнтів, щоб проаналізувати, як вони себе почувають під час госпіталізації або перед випискою;

– індустрія безпеки. Ця нова технологія грає одну з життєво важливих ролей в індустрії безпеки. Будь то офіс, смартфон, банк або будинок, заходи безпеки є невід'ємною частиною багатьох платформ. Було розроблено безліч пристроїв безпеки, включаючи дрони, камери безпеки, біометричні пристрої для розпізнавання осіб і т. д. На виставці CES 2019 були представлені камери домашньої безпеки SimCam і домашньої автоматизації, оснащені штучним інтелектом для розпізнавання осіб, спостереження за тваринами і багато чого іншого. Netatmo, розумна

кімнатна камера, має функцію, яка починає запис відео тільки тоді, коли система виявляє в приміщенні будь-яку невідому особу;

– роздрібна торгівля. У роздрібній торгівлі існує величезний попит на цей революційний метод. Наприклад, Trax призначений для роздрібної торгівлі та споживчих товарів, які містять базу даних SKU, обробляючи 40 000 зображень на годину. Якість і ціну продукту можна порівняти за допомогою методу розпізнавання зображень. Tesco працює з біометричними технологіями і технологіями розпізнавання зображень для численних механізмів, що охоплюють розпізнавання, геометрію, оцінку якості для створення технологій для магазинів, планування асортименту продуктів і багато чого іншого;

– соціальні мережі. Розпізнавання зображень досить добре працює в цій галузі, оскільки маркетологам стало простіше знаходити графічні матеріали в соціальних мережах. Інструменти розпізнавання зображень можуть шукати зображення в соціальних мережах і порівнювати їх з великими базами даних, щоб знаходити потрібні зображення з безпрецедентною швидкістю і масштабом. У 2016 році Facebook додав функцію для людей з ослабленим зором, об'єднавши техніку розпізнавання осіб і автоматичні текстові технології для створення точного опису змісту фотографії, а також опису того, хто саме знаходиться на фотографії без позначки;

– системи візуального пошуку. Дана технологія використовує розпізнавання зображень, щоб надати користувачам кращі результати пошуку. Google і Bing є найбільшими представниками цієї платформи, але є й інші системи візуального пошуку. Наприклад, Picsearch – це традиційна система візуального пошуку, яка пропонує величезний архів зображень;

– змішана реальність. Це технологія поєднання віртуальної реальності і доповненої реальності. Щоб подолати недоліки VR і AR, створюється змішана реальність, яка забезпечує більш динамічний і природний досвід віртуального світу. Наприклад, Intel Project Alloy або

Microsoft Windows Holographic Shell, бездротова гарнітура, яка дозволяє переносити реальні об'єкти в віртуальний світ за допомогою 3D-камер.

1.3 Огляд існуючих методів розв’язання задачі розпізнавання

З різноманіття типів нейронних мереж, спрямованих на вирішення різних завдань, можна виділити кілька, переважно які спеціалізуються на завданнях роботи з образами і зображеннями. Далі будуть стисло розглянуті найбільш широко використовувані з них.

В останнє десятиліття згорткові нейронні мережі (CNN) стали одним з найпопулярніших методів вирішення проблем комп'ютерного зору. Численні завдання зору, такі як класифікація зображень, виявлення об'єктів, розпізнавання обличь, отримали користь від надійного та дискримінаційного уявлення, отриманого через CNN.

Нижче ми розглянемо найсучасніші та найвідоміші архітектури, які були розроблені та успішно реалізовані в різних сферах обробки зображень та розпізнавання об'єктів.

LeNet-5. LeNet-5 широко використовувалася для автоматичної класифікації рукописних цифр на банківських чеках у США. Ця мережа є згортковою нейронною мережею (CNN). CNN є основою сучасного комп’ютерного зору на основі глибинного навчання. Ці мережі побудовані на трьох основних ідеях: місцеві рецептивні поля, спільні ваги та просторова субдискретизація. Місцеві рецептивні поля зі спільними вагами є суттю згорткового шару, і більшість описаних нижче архітектур використовують згорткові шари в тій чи іншій формі.

За сучасними стандартами LeNet-5 – це дуже проста мережа. Вона має лише 7 шарів, серед яких є 3 згорткові шари (C1, C3 та C5), 2 шари субвибірки (об'єднання) (S2 та S4) та 1 повністю зв’язаний шар (F6), за якими слідує вихід шар (рисунок 1.2). Згорткові шари використовують згортки 5x5 з кроком 1. Шари підвибірок – це 2x2 середні шари об'єднання.

Сигмоїдальна функція активації використовуються у всій мережі. Існує кілька цікавих архітектурних рішень, зроблених у LeNet-5, які не дуже поширені в сучасну епоху глибинного навчання.

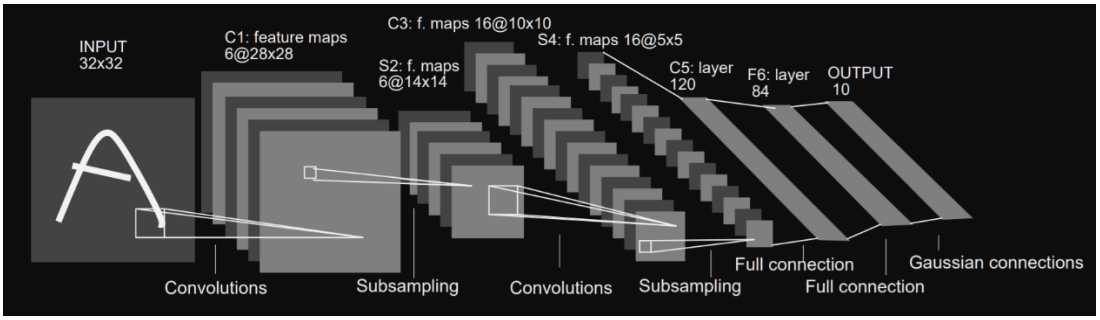

Рисунок 1.2 – Архітектура LeNet-5

По-перше, окремі згорткові ядра в шарі C3 використовують не всі ознаки, що створюються шаром S2, що є дуже незвичним за сучасними стандартами. Одна з причин цього – зробити мережу менш вимогливою до обчислень. Іншою причиною було змусити згорткові ядра вивчати різні моделі. Це цілком логічно: якщо різні ядра отримують різні вхідні дані, вони вивчать різні моделі.

По-друге, вихідний шар використовує 10 нейронів Евклідової функції радіального базису, які обчислюють відстань L2 між вхідним вектором розмірності 84 та заздалегідь заданими ваговими векторами тієї самої розмірності. Число 84 походить від того, що ваги представляють двійкову маску 7x12, по одній для кожної цифри. Це змушує мережу трансформувати вхідне зображення у внутрішнє представлення, яке зробить виходи шару F6 максимально наближеними до кодованих вручну ваг 10 нейронів вихідного шару.

Це був один із успішних алгоритмів розпізнавання свого часу, реалізованих для класифікації рукописних цифр. Сьогоднішня реалізація цієї архітектури на наборах даних із використанням різних бібліотек дозволить отримати точність близько 98,9%.

Однак, коли мова заходила про обробку зображення великого розміру та класифікацію серед великої кількості об’єктів, ця мережа не могла бути ефективною з точки зору вартості обчислення або точності.

AlexNet. Мережа має дуже схожу архітектуру з LeNet, але є глибшою, з більшою кількістю фільтрів на шар і з накопиченими згортковими шарами. Архітектура складається з восьми шарів: п'яти згорткових шарів та трьох повністю з'єднаних шарів (рисунок 1.3).

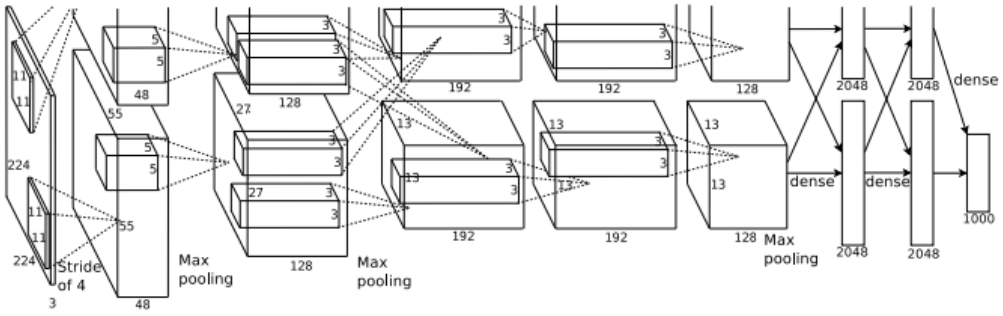

Рисунок 1.3 – Архітектура AlexNet

Особливості AlexNet:

– нелінійність ReLU. AlexNet використовує випрямлені лінійні одиниці (ReLU) замість стандартної на той час функції гіперболічного тангенсу. Перевага ReLU полягає у навчальному часі; CNN, яка використовує ReLU, змогла досягти 25% помилки в наборі даних CIFAR-10 у шість разів швидше, ніж CNN на функції гіперболічного тангенсу;

– навчання на декількох GPU. AlexNet може навчатися на декількох графічних процесорах, розміщуючи половину нейронів моделі на одному графічному процесорі, а другу половину – на іншому графічному процесорі. Це не тільки означає, що більшу модель можна навчити, але й скорочує час навчання;

– overlapping Pooling. CNN традиційно «об'єднують» виходи сусідніх груп нейронів без перекриття. Однак, Overlapping Pooling зменшує кількість помилок приблизно на 0,5%;

– проблема перенавчання. AlexNet має 60 мільйонів параметрів. Для зменшення перенавчання використовуються два методи;

– data augmentation. Використання трансформації для збільшення різноманітності даних. Зокрема, генеруються обертання зображень та горизонтальні відбиття, що збільшує навчальний набір в тисячі разів;

– dropout. Ця методика складається з «вимкнення» нейронів із заздалегідь визначеною ймовірністю (наприклад, 50%). Це означає, що кожна ітерація використовує різні вибірки параметрів моделі, що змушує кожен нейрон мати більш надійні функції, які можна використовувати з іншими випадковими нейронами. Однак відсівання також збільшує час навчання, необхідний для зближення моделі.

VGGNet. Повна назва VGG – Visual Geometry Group. VGG можна застосовувати для розпізнавання обличчя та класифікації зображень, починаючи з VGG16 і закінчуючи VGG19. Початковою метою досліджень VGG щодо глибини згорткових мереж є розуміння, як глибина згорткових мереж впливає на точність масштабної класифікації та точність розпізнавання зображень. Щоб поглибити кількість шарів і уникнути занадто великої кількості параметрів, у всіх шарах використовується невелике ядро згортки 3x3.

Для вхідних даних VGG встановлено зображення RGB розміром 224x244. Середнє значення RGB обчислюється для всіх зображень на зображенні навчального набору, а потім зображення вводиться як вхід до мережі згортки VGG. Використовується фільтр 3x3 або 1x1, і крок згортки фіксований. Існує 3 повністю з'єднаних шари VGG, які можуть варіюватися від VGG11 до VGG19 відповідно до загальної кількості згорткових шарів + повністю з'єднаних шарів. Мінімальний VGG11 має 8 згорткових шарів та 3 повністю з'єднаних шари. Максимум

VGG19 має 16 згорткових шарів та 3 повністю з'єднаних шарів. Крім того, за мережею VGG не супроводжується шар об'єднання позаду кожного згорткового шару, або загалом 5 шарів об'єднання, розподілених під різними згортковими шарами. Архітектура VGG представлена на рисунку 1.4.

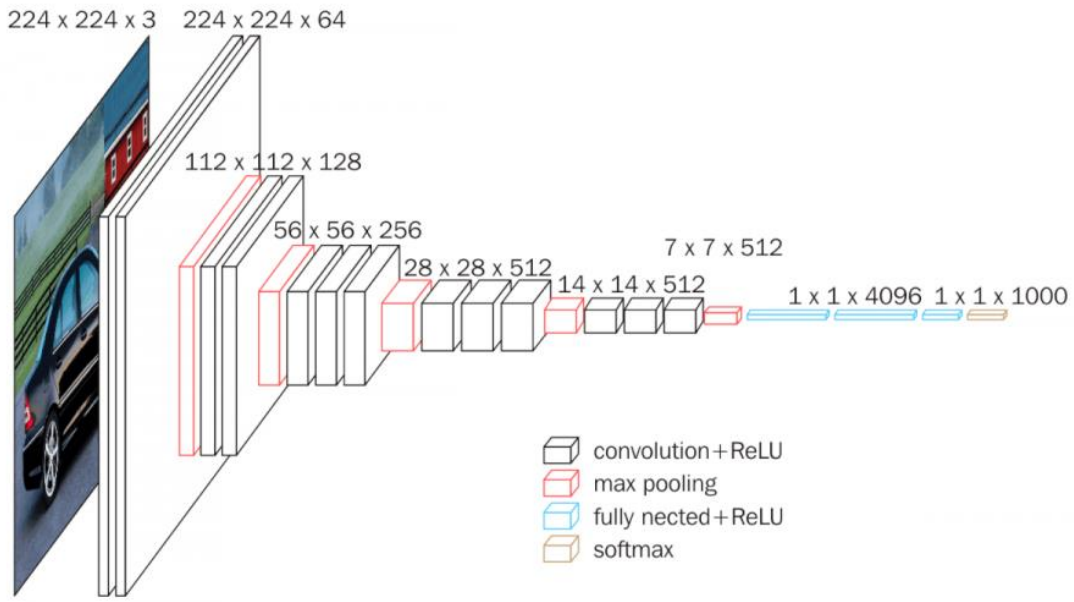

Рисунок 1.4 – Архітектура VGGNet

VGG16 містить 16 шарів, а VGG19 – 19 шарів. Серія VGG абсолютно однакова в останніх трьох повністю з'єднаних шарах. Загальна структура включає 5 наборів згорткових шарів, за якими йде MaxPool. Різниця полягає в тому, що все більше і більше каскадних згорткових шарів включаються до п’яти наборів згорткових шарів.

Наприклад кожен згортковий шар в AlexNet містить лише одну згортку, а розмір ядра згортки становить 7×7,. У VGGNet кожен рівень згортки містить від 2 до 4 операцій згортки. Розмір ядра згортки – 3×3, розмір кроку згортки – 1, ядро об'єднання – 2×2, а розмір кроку – 2. Найбільш очевидним вдосконаленням VGGNet є зменшення розміру ядра згортки та збільшення кількість згорткових шарів.

Дві послідовні 3×3 згортки еквівалентні 5×5 рецептивному полю, а три еквівалентні 7×7. Переваги використання трьох 3×3 згорток замість однієї 7×7 подвійної згортки наступні: використання трьох шарів ReLu замість одного, що робить функцію прийняття рішень більш характерною; зменшена кількість параметрів.

GoogLeNet/Inception. Архітектура GoogLeNet сильно відрізняється від попередніх сучасних архітектур, таких як AlexNet та ZF-Net. Вона використовує багато різних методів, таких як згортка 1×1 та середнє глобальне об'єднання, що дозволяє створити більш глибинну архітектуру.

Згортка 1×1. Початкова архітектура використовує згортку 1×1 у своїй архітектурі. Ці згортки використовувались для зменшення кількості параметрів (ваг та зміщень) архітектури. Зменшуючи параметри, ми також збільшуємо глибину архітектури. На рисунку 1.5 зображено приклад згортки 1×1.

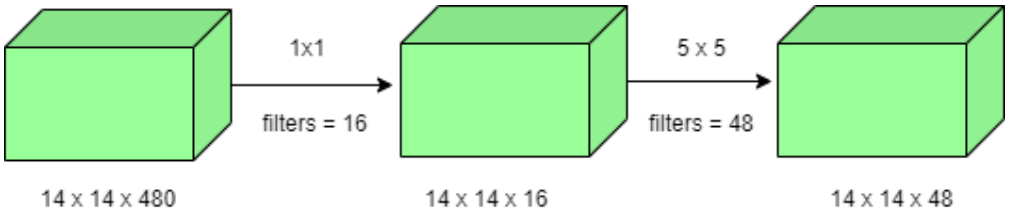

Рисунок 1.5 – Приклад згортки з використанням проміжної згортки 1×1

Глобальне середнє об'єднання. У попередній архітектурі, такій як AlexNet, повноз'єднані шари використовуються в кінці мережі. Ці шари містять більшість параметрів багатьох архітектур, що спричиняє збільшення вартості обчислень.

В архітектурі GoogLeNet існує метод, який називається глобальним середнім пулом (Global Average Pooling), який використовується в кінці мережі. Цей шар приймає карту функцій 7×7 і в середньому

становить 1×1. Це також зменшує кількість параметрів, що піддаються підготовці, до 0 і покращує точність на 0,6%.

Модуль утворення. Модуль утворення відрізняється від попередніх архітектур, таких як AlexNet, ZF-Net. У цій архітектурі існує фіксований розмір згортки для кожного шару.

У цьому модулі 1×1, 3×3, 5×5 згортки та максимальне об’єднання 3×3 виконуються паралельно на вході та на виході з них, та складаються разом для отримання кінцевого результату. Ідея полягає в тому, що фільтри згортки різного розміру краще оброблятимуть об'єкти в більшому масштабі. Рисунок 1.6 зображує схему модуля утворення.

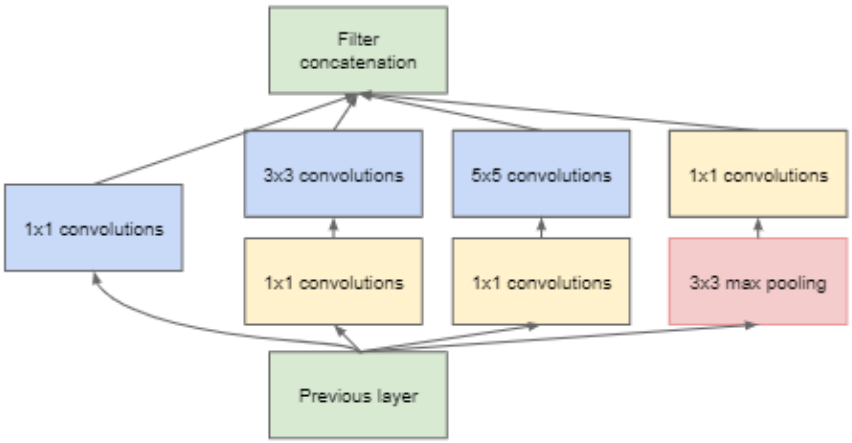

Рисунок 1.6 – Модуль утворення зі зменшенням розмірів

Допоміжний класифікатор для навчання. Початкова архітектура використовувала деякі проміжні ребра класифікатора в середині архітектури, ці ребра використовуються лише під час навчання. Вони складаються з 5×5 середнього шару об’єднання з кроком 3, згортки 1×1 із 128 фільтрами, двох повноз’єднаних шарів з 1024 та 1000 виходами та шару класифікації softmax. Генеровані втрати цих шарів додаються до загальних втрат з вагою 0,3. Ці шари допомагають боротися з проблемою зникнення градієнта, а також забезпечують регуляризацію.

Загальна архітектура нараховує 22 шари. Вона була розроблена з урахуванням ефективності обчислень, тобто задля того, щоб могла запускатися на окремих пристроях з малою кількістю обчислювальних ресурсів. Архітектура також містить два допоміжні шари класифікатора, з'єднані з вихідними рівнями. На рисунку 1.7 зображена повна архітектура мережі GoogLeNet.

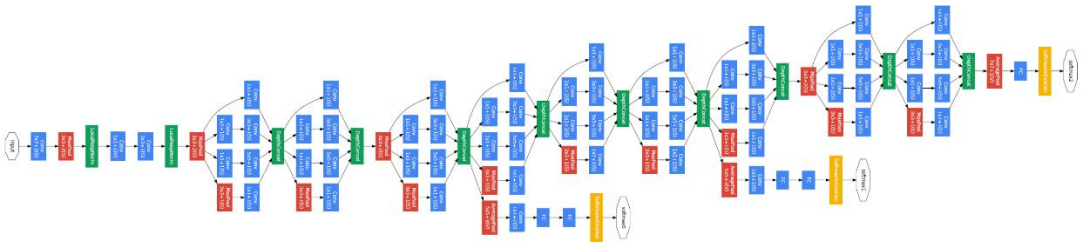

Рисунок 1.7 – Загальна архітектура GoogLeNet

ResNet. Відповідно до теореми універсального наближення, маючи достатню пропускну здатність, ми знаємо, що прямої мережі з одним шаром достатньо для представлення будь-якої функції. Однак шар може бути масивним і мережа схильна до перенавчання даних. Тому в науковому співтоваристві існує загальна тенденція, згідно з якою наша мережева архітектура повинна заглиблюватися.

Починаючи з AlexNet, надсучасна архітектура CNN стає все глибшою та глибшою. Хоча AlexNet мав лише 5 згорткових шарів, мережа VGG [1] та GoogleNet [2] мали 19 і 22 шари відповідно.

Однак збільшення глибини мережі не працює, просто складаючи шари разом. Глибинні мережі важко тренувати через проблему зникаючого градієнта – оскільки градієнт знову поширюється на попередні шари, багаторазове множення може зробити градієнт нескінченно малим. Як результат, із поглибленням мережі її ефективність починає швидко погіршуватися.

Основною ідеєю ResNet є введення «з'єднання швидкого доступу».

Пояснювальна схема з'єднання швидкого доступу зображена на рисунку 1.8.

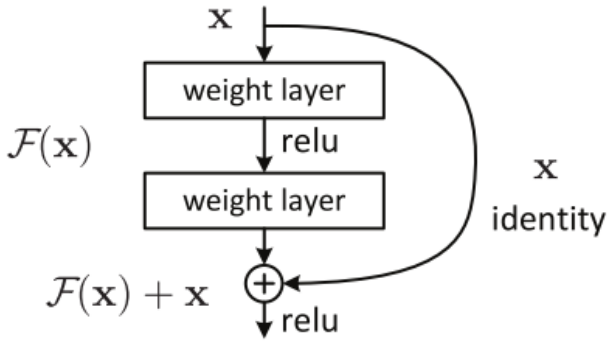

Рисунок 1.8 – Приклад з'єднання швидкого доступу

З'єднання швидкого доступу (shortcut connections) пропускають один або кілька шарів і виконують зіставлення ідентифікаторів. Їх виходи додаються до виходів stacked layers. Використовуючи ResNet, можна вирішити безліч проблем, таких як:

– resnet відносно легко оптимізувати: «прості» мережі (які просто складають шари) показують велику помилку навчання, коли глибина збільшується;

– resnet дозволяє відносно легко збільшити точність завдяки збільшенню глибини, чого з іншими мережами домогтися складніше;

– архітектура ResNet заснована на архітектурі простої мережі з доповненням з'єднання швидкого доступу, яке перетворює мережу в її остаточну версію;

– з'єднання швидкого доступу $F(x\{W\} + x)$ може використовуватися безпосередньо, коли вхід і вихід мають однакові розмірності. Коли розмірності збільшуються, розглядається два варіанти;

– з'єднання виконує зіставлення ідентифікаторів з додатковими нулями, доданими для збільшення розмірності. Ця опція не вводить ніяких додаткових параметрів;

– проекція швидкого з'єднання в $F(x\{W\}+x)$ використовується для зіставлення розмірності (виконано за допомогою згортки 1×1).

Кожен блок ResNet має два рівня глибини (використовується в невеликих мережах, таких як ResNet 18, 34) або 3 рівня (ResNet 50, 101, 152).

50-шарова ResNet. Кожен 3-шаровий блок замінюється в 34-шаровій мережі цим 3-шаровим вузьким місцем, в результаті виходить 50-шарова ResNet.

ResNet з 101 і 152 шарами. Створюється ResNet з 101 і 152 шарами, використовуючи більше 3-шарових блоків. Навіть після збільшення глибини 152-шарова ResNet (11,3 мільярда FLOP) має меншу складність, ніж мережі VGG-16/19 (15,3 / 19,6 мільярда FLOPs).

1.4 Виявлення проблем в задачі розпізнавання

Розпізнавання зображень має стикатися з неконтрольованими умовами освітлення, великими варіаціями точок зору, варіаціями розміру та оклюзіями. Існує декілька проблем і ключових факторів, які можуть суттєво вплинути на продуктивність розпізнавання, а також інші фактори, які можуть вплинути на відповідність результатів.

Також системи розпізнавання потребують у тисяч разів більше даних ніж люди, результат їх роботи залежить від даних. Глибинне навчання спирається на величезні обсяги високоякісних вибірок для прогнозування майбутніх тенденцій і побудови моделей класифікації. Часто при підготовці системи розпізнавання дуже важко зібрати дані для навчання, у яких усі класи представлені в однаковому та достатньому об’ємі. Набори даних повинні бути репрезентативними для класів, які ми маємо намір передбачити, в іншому випадку система буде узагальнювати асиметричний розподіл класів, а зміщення буде руйнувати модель.

Нижче проілюстровані та описані головні фактори, які можуть суттєво вплинути на точність та якість розпізнавання.

Освітлення (Illumination). Коли формується зображення, такі фактори як освітлення (розподіл джерел і інтенсивність) і характеристики камери (потужність датчика і лінзи) впливають певною мірою на вигляд об’єкту розпізнавання. Зміни освітлення можуть також впливати через властивості відбиття поверхні та через внутрішній контроль камери, як показано на рисунку 1.9. При побудові надійної та ефективної системи розпізнавання проблема зміни освітлення вважається одним із головних технічних завдань, що стоїть перед розробниками системи. Кілька двовимірних методів добре виконують завдання розпізнавання тільки при помірній варіації освітлення, тоді як продуктивність помітно падає при яскравому освітленні.

Незважаючи на те, що методи попередньої обробки зображень є простими і швидкими, вони ігнорують вплив змін напрямку освітлення, що призводить до великих змін в локальних характеристиках.

Рисунок 1.9 – Варіації освітлення одного об’єкту

Варіація точки зору (Viewpoint Variation). Об’єкт може бути орієнтований або повернутий у різних вимірах залежно від того, як об’єкт фотографується та фіксується на зображенні (рисунок 1.10). Також деякі значимі частини об’єкту можуть частково або повністю закриватися.

Таким чином, позиція об’єкту розпізнавання на зображенні стає ще більш критичною для систем розпізнавання, які спираються на єдиний погляд суб'єкта. Для вирішення даної проблеми застосовуються алгоритми перетворення або деформування зображень в процесі попередньої обробки, на основі оцінених параметрів у відповідності до вивченої вибірки.

Рисунок 1.10 – Варіації точки зору одного об’єкту

Видимість об’єкту (Occlusion). У зображенні з групою об’єктів об’єкт розпізнавання може бути частково закритий, що в свою чергу призводить до ситуації, коли доступна лише невелика його частина (рисунок 1.11). Це робить завдання розпізнавання складнішим, і навіть якщо об’єкт знайдено, то деякі його важливі частини можуть бути скритими, що робить неможливим подальше розпізнавання.

Ця проблема може бути вирішена за допомогою навчання на додаткових вибірках, де об’єкти є частково прихованими. При такому підході важливо пам’ятати, що система розпізнавання повинна і надалі вилучати точні і дискриміновані ознаки, які зменшують схожість між різними об’єктами.

Рисунок 1.11 – Приклад оклюзії об’єкту розпізнавання

Внутрішньокласові варіації (Intra-Class Variation). Істотні внутрікласові варіації, викликані різними модальностями візуалізації і різними типами ілюстрацій, вносять великі труднощі в проблему розпізнавання. Приклад внутрішньокласових варіацій показаний на рисунку 1.12.

Рисунок 1.12 – Приклад внутрішньокласових варіацій

Хоча глибинні нейронні мережі мають достатню здатність примусово запам'ятовувати всі навчальні вибірки, неоднозначність, викликана внутрікласовими варіаціями, може збити з пантелику мережу, і, як результат, вона приймає правильне рішення з низьким рівнем впевненості, і результати можуть навіть бути абсолютно протилежними, якщо додати невеликі коливання у вхідні дані.

Варіації розміру (Scale Variation). Згорткові нейронні мережі (CNN) вимагають підготовки великих масивів зображень для задач класифікації. Різниця в роздільній здатності зображень, розмірах зображених предметів, візерунків та масштабах зображень заважає навчанню та продуктивності мереж, оскільки інформація, варіюється в просторових масштабах (рисунок 1.13).

Рисунок 1.13 – Приклад варіації розміру

Згорткові фільтри, які автоматично налаштовуються під час навчальної процедури CNN, зазвичай вирішують проблему з цими варіаціями. CNN повинна ігнорувати нерелевантні варіації роздільної здатності зображення, розміру об'єкта, масштабу зображення та враховувати відповідні функції задачі в конкретному масштабі. Фільтри,

що забезпечують таку підтримку, називаються, інваріантно-масштабними та масштабними фільтрами відповідно.

Перешкоди на задньому фоні (Background Clutter). Background Clutter означає, що на зображенні є багато об’єктів, і для спостерігача дуже важко знайти конкретний. Ці зображення також називають «шумними» (рисунок 1.14).

В деяких випадках на зображенні є певний об'єкт, зовнішній вигляд якого дуже схожий на фон, тому в цих випадках системі розпізнавання дуже складно класифікувати його.

Рисунок 1.14 – Приклад перешкод, шуму на задньому фоні

1.5 Постановка задачі

Розпізнавання зображень являє собою складну задачу інтелектуального характеру, для якої неможливо створити універсальне рішення. На даний час існує велика кількість підходів, щодо вирішення цього завдання.

Метою роботи є розробка обчислювально-ефективної архітектури глибинної нейронної мережі для розпізнавання статичних зображень.

Для досягнення поставленої мети необхідно розглянути наступні питання:

– провести аналіз існуючих методів для вирішення задачі розпізнавання;

– проаналізувати проблеми вирішення задачі розпізнавання і визначити шляхи їх вирішення;

– розробити архітектуру глибинної згорткової нейронної мережі;

– провести тестування та порівняльний аналіз розробленої системи розпізнавання з існуючими на даний момент методами.

2 ОГЛЯД ІСНУЮЧИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ

2.1 Нейронні мережі

Нейронні мережі, також відомі як штучні нейронні мережі (ШНМ) або модельовані нейронні мережі (МНМ) є підмножиною машинного навчання і лежать в основі алгоритмів глибинного навчання. Їх назва та архітектура натхненні людським мозком, вони імітують спосіб передачі сигналів від одного до другого нейрону мозку.

Штучні нейронні мережі складаються з вузлових шарів, що містять вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен вузол або штучний нейрон з'єднується з іншим і має вагу та поріг. Якщо вихід будь-якого окремого вузла вище зазначеного порогового значення, цей вузол активується, відправляючи дані на наступний рівень мережі. В іншому випадку на наступний рівень мережі ніякі дані не передаються. Приклад архітектури нейронної мережі представлений на рисунку 2.1.

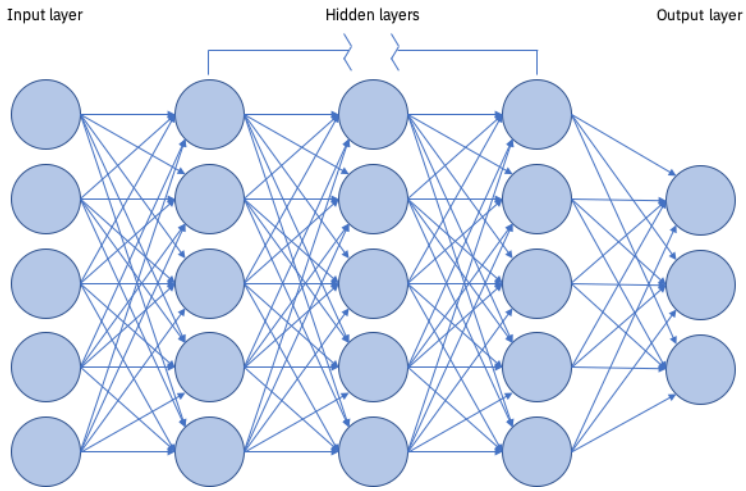

Рисунок 2.1 – Приклад архітектури штучної нейронної мережі

Проектування вхідного і вихідного рівнів в мережі часто буває простим. Наприклад, потрібно визначити, чи є рукописна «9» на зображенні чи ні. Очевидний спосіб спроектувати мережу – це кодувати інтенсивність пікселів зображення у вхідних нейронах. Якщо зображення представлене в градаціях сірого розміром 64x64, то у нас буде $4096 = 64 \times 64$ вхідних нейронів з відповідними масштабами інтенсивності від 0 до 1. Вихідний шар міститиме тільки один нейрон з вихідними значеннями менше 0,5, що вказує на «вхідне зображення – не 9», і значення більше 0,5, що вказує «вхідне зображення – 9».

Хоча проектування вхідних і вихідних шарів нейронної мережі часто буває простим, створення прихованих шарів може бути справжнім викликом. Зокрема, неможливо підвести підсумок процесу проектування прихованих шарів за допомогою декількох простих практичних правил. Замість цього дослідники нейронних мереж розробили безліч евристик проектування для прихованих шарів, які допомагають добитися від мереж бажаної поведінки. Наприклад, таку евристику можна використовувати для визначення того, як знайти компроміс між кількістю прихованих шарів і часом, необхідним для навчання мережі.

Вище описувалися нейронні мережі, де вихідні дані одного шару використовуються як вхідні дані для наступного шару. Такі мережі називаються нейронними мережами прямого поширення. Це означає, що в мережі немає циклів – інформація завжди прямує вперед та ніколи назад. Приклад мережі прямого поширення представлений нижче на рисунку 2.2.

Нейронні мережі прямого поширення в основному використовуються у навчанні з вчителем в тих випадках, коли дані, що підлягають засвоєнню, не є ані послідовними, ані залежними від часу. Тобто такі нейронні мережі обчислюють функцію f на введеному фіксованому розмірі x таким, що $f(x) \approx y$ для тренувальних пар (x, y).

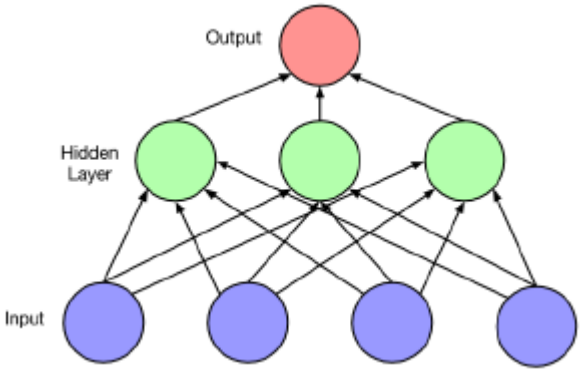

Рисунок 2.2 – Нейронна мережа прямого поширення

Вхід x до мережі прямого поширення, забезпечується шляхом встановлення значень нижчого рівня (рисунок 2.2). Кожен вищий рівень послідовно обчислюється, поки не буде отриманий вихід на верхньому шарі y. Для вирішення завдань класифікації та регресії часто використовуються мережі прямого поширення. Навчання здійснюється шляхом ітеративного оновлення ваги на кожному вузлу задля мінімізації функції втрат, $\mathcal{L}(\hat{y}, y)$, яка зменшує відстань між виходом $\hat{y}$ і цільовим y.

Як правило, процес навчання вимагає визначення функції помилки E, яка кількісно визначає різницю між виходом нейрона o і справжнім значенням y для вхідного сигналу $\vec{x}$ над набором безлічі пар входу-виходу (x, y). Історично ця функція помилки є середньоквадратичною помилкою, визначеною як вхід-вихід N $X = \{(x1, y1), \dots, (xN, yN)\}$ як:

$$E(X) = \frac{1}{2N}\sum_{i=1}^{N}(o_i - y_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}(g(\vec{w} \cdot \vec{x_i} + b) - y_i)^2, \qquad (2.1)$$

де o_i – позначає вихід нейрона;

$\vec{x_i}$ – вхідні нейрони;

g – функція активації.

Наступний вид нейронних мереж – це нейронні мережі зворотного поширення. Зворотне поширення відноситься до методу розрахунку градієнта параметрів нейронної мережі.

Принцип підходу зворотного поширення полягає в моделюванні заданої функції шляхом модифікації внутрішніх ваг вхідних сигналів для отримання очікуваного вихідного сигналу. Для тренування використовується підхід навчання з вчителем, де помилка між результатом роботи системи та відомим очікуваним результатом використовується для модифікації її внутрішнього стану.

Технічно алгоритм зворотного поширення помилки – метод проходить по мережі в зворотному порядку, від вихідного до вхідного рівня, відповідно до правила ланцюга з обчислення. Алгоритм зберігає будь-які проміжні змінні (приватні похідні), необхідні при обчисленні градієнта за деякими параметрами. Припустимо, що у нас є функції $Y = f(X)$ та $Z = g(Y)$, в яких вхід и вихід – X, Y, Z є багатовимірним масивом даних. Використовуючи правило ланцюга, ми можемо обчислити похідну від Z відносно X використовуючи наступну формулу:

$$\frac{\partial Z}{\partial X} = prod\left(\frac{\partial Z}{\partial Y}, \frac{\partial Y}{\partial X}\right), \tag{2.2}$$

де X, Y, Z – багатовимірні масиви.

Тут ми використовуємо оператор $prod$ для множення аргументів після того, як були проведені необхідні операції, такі як транспонування та зміна входів. У разі векторів просто потрібно перемножити матриці. У разі багатовимірних масивів вищих розмірів ми використовуємо відповідний аналог.

$W^{(1)}$ та $W^{(2)}$ є параметрами простої мережі з одним прихованим шаром. Метою зворотного поширення є розрахунок градієнтів $\frac{\partial J}{\partial W^{(1)}}$ та

$\frac{\partial J}{\partial W^{(2)}}$. Для цього застосовується правило ланцюга і обчислюється у свою чергу градієнт кожної проміжної змінної та параметра. Порядок обчислень зворотній порівняно з тими, що виконуються при прямому поширенні, оскільки потрібно починати з результату обчислювального графіку і рухатися до параметрів.

Однак є й інші моделі штучних нейронних мереж, в яких можливі петлі зворотного зв'язку. Ці моделі називаються рекурентними нейронними мережами (RNN). Ідея цих моделей полягає в тому, що нейрони спрацьовують протягом деякого обмеженого періоду часу, перш ніж стати спокійними. Це збудження може стимулювати інші нейрони, які можуть активуватися трохи пізніше, але також на обмежений час. Це викликає спрацьовування ще більшої кількості нейронів, і з часом ми отримуємо каскад спрацьовувань. Цикли не викликають проблем в такій моделі, адже вихід нейрона впливає на його вхід тільки в більш пізній час, а не миттєво. На рисунку 2.3 зображена рекурентна нейронна мережа.

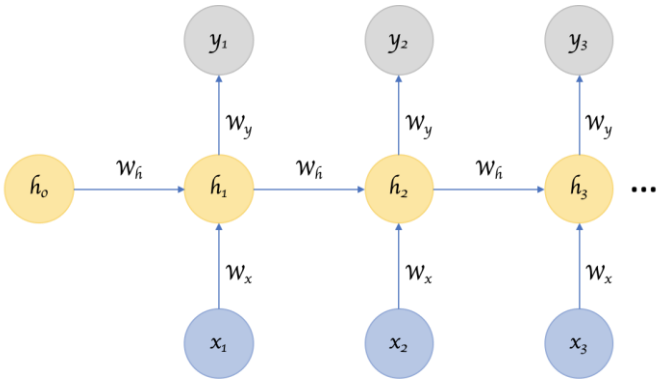

Рисунок 2.3 – Рекурентна нейронна мережа з прихованим станом

RNN можуть приймати один або кілька вхідних векторів і створювати один або кілька вихідних векторів, і на вихід (виходи) впливають не тільки ваги, що застосовуються до входів, як у звичайній штучній мережі, але також «прихований» вектор стану, що представляє

контекст на основі попередніх входів / виходів. Таким чином, один і той же вхід може давати різні результати на виході в залежності від попередніх входів в серії.

Рекурентні нейронні мережі були менш впливовими, ніж мережі прямого поширення, частково тому що алгоритми навчання для рекурентних мереж (принаймні, на сьогоднішній день) менш потужні. Але рекурентні мережі як і раніше вкрай цікаві. Їх архітектура більш схожа на те, як працює мозок людини, ніж мережі прямого поширення. І можливо такі мережі розв’яжуть важливі проблеми, які на даний момент з великими труднощами можуть бути вирішені тільки мережами прямого поширення.

2.2 Згорткові операції

Згорткова операція – це математична операція двох функцій, яка на виході результує нову функцію. Якщо графіки оригінальних функцій збігаються, то нова функція покаже міру подібності.

Теорема про згортку стверджує, що за певних умов перетворення згортки Фур'є є елементом перетворення Фур'є. Іншими словами, згортка в одному домені дорівнює добутку елементів в іншому домені.

В одновимірному просторі згортка між двома функціями визначається наступним чином:

$$g(x) = f(x) \odot h(x) = \int_{-\infty}^{\infty} f(s)\, h(x - s)ds, \qquad (2.3)$$

де $f(x)$ та $h(x)$ вихідні функції;

s – змінна інтеграції (приймає значення від 0 до 1).

В двовимірному просторі операція згортання між двома функціями визначається наступним чином:

$$g(x,y) = f(x,y) \odot h(x,y) = \int\int_{-\infty}^{\infty} f(s,t)h(x-s,y-t)\,ds\,dt, \qquad (2.4)$$

де $f(x)$ та $h(x)$ вихідні функції;

s – змінна інтеграції (приймає значення від 0 до 1).

Операція згортки зазвичай позначається зірочкою *.

Наприклад, в одновимірних додатках ми маємо часову область сигналу $x(t)$ і частотну область $w(a)$. Керуючись теоремою згортки, операція згортки буде мати наступний вигляд (формула 2.9) [6]:

$$s(t) = (x * w)(t), \qquad (2.5)$$

де x є входом;

w – ядро;

$s(t)$ – карта ознак або карта ядра.

У комп'ютерних додатках дані часових рядів будуть дискретизовані, а індекс часу t може приймати тільки цілі значення. Таким чином, дискретна згортка може бути визначена за формулою:

$$s(t) = (x * w)(t) = \sum_{-\infty}^{\infty} x(a)w(t-a), \qquad (2.6)$$

де $s(t)$ карта ознак або карта ядра;

x – вхід;

w – ядро;

t – час.

У додатках з машинним навчанням вхід, як правило, є багатовимірним масивом даних, тому ядро зазвичай є багатовимірним

масивом параметрів, які адаптуються алгоритмом навчання [6]. Ці багатовимірні масиви називаються тензорами.

Якщо є сигнал у двовимірному просторі, наприклад, зображення, то необхідно використовувати двомірне ядро K. Згортка для двох вимірів полягає в наступному:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \qquad (2.7)$$

де $S(i,j)$ – вихід;

I – рецептивне поле;

K – ядро;

m – поточне значення довжини;

n – поточне значення ширини;

i – довжина;

j – ширина.

Якщо припустити, що в діапазоні допустимих значень m порівняно з n, на підставі припущення, що згортка може бути переміщена, ми можемо еквівалентно записати (2.7) наступним чином:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \qquad (2.8)$$

де $S(i,j)$ – вихід;

I – рецептивне поле;

K – ядро;

m – поточне значення довжини;

n – поточне значення ширини;

i – довжина;

j – ширина.

У випадку, якщо m зростає, то індекс у вхідному сигналі також зростає, але індекс у ядрі зменшується. Це означає, що ядро перевернуте відносно входу. Якщо ядро не перевернуто, використовується пов'язана функція, що називається крос-кореляцією:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n), \qquad (2.9)$$

де $S(i,j)$ – вихід;

I – рецептивне поле;

K – ядро;

m – поточне значення довжини;

n – поточне значення ширини;

i – довжина;

j – ширина.

У контексті машинного навчання алгоритм навчається знаходити відповідні значення ядра у відповідному місці [6]. У машинному навчанні згортка не використовується окремо, а одночасно з поєднанням інших функцій. На основі принципів згортки працюють згорткові мережі (CNN).

2.3 Згорткові нейронні мережі

Архітектура CNN натхненна відкриттям механізму зорової кори головного мозку. Зорова кора містить безліч клітин, які відповідають за виявлення світла, які називаються рецептивними полями. Ці клітини діють як локальні фільтри у вхідному просторі, а більш складні – мають більші рецептивні поля. Згортковий шар в CNN виконує функцію, яку виконують клітини зорової кори [1].

Типова CNN для розпізнавання дорожніх знаків показана на рисунку 2.4. Кожна функція шару отримує вхідні дані від набору функцій,

розташованих в невеликому районі на попередньому шарі, званому локальним рецептивним полем. За допомогою локальних рецептивних полів об'єкти можуть знаходити елементарні візуальні ознаки, такі як орієнтовані краї, кінцеві точки, кути контурів і т. д., які потім об'єднуються більш високими рівнями.

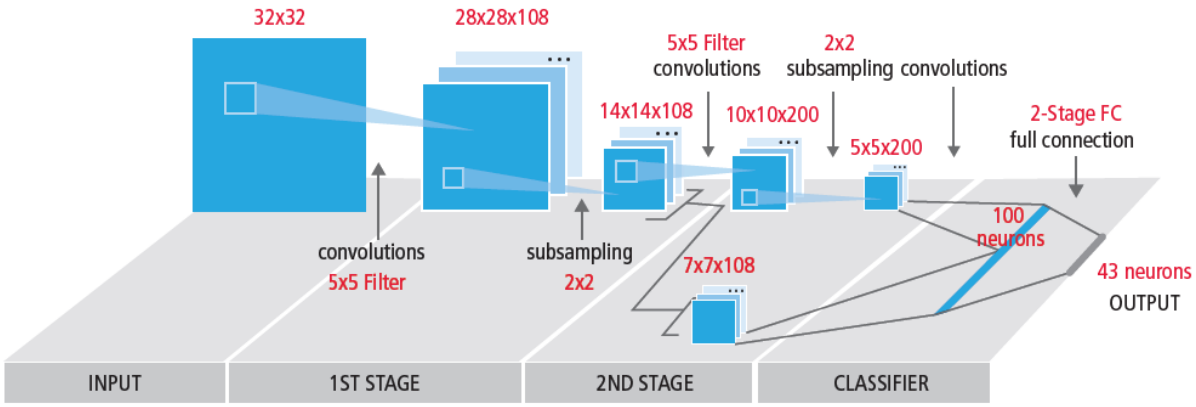

Рисунок 2.4 – Типова архітектура CNNs

У традиційній моделі розпізнавання образів, створений вручну екстрактор ознак збирає релевантну інформацію з вхідних даних і усуває несуттєві відхилення. За екстрактором розміщується класифікатор – стандартна нейронна мережа, яка класифікує вектори ознак за класами.

У CNN згорткові шари грають роль екстрактора ознак. Але вони не створені вручну. Ваги ядра фільтра згортки визначаються в процесі навчання. Згорткові шари можуть витягувати локальні особливості, так як вони обмежують рецептивні поля.

CNN використовуються в різних областях, включаючи розпізнавання зображень і образів, розпізнавання мови і аналіз відео. Існує ряд причин, за якими згорткові нейронні мережі стають важливими. У традиційних моделях розпізнавання образів екстрактори ознак розробляються вручну. У CNN ваги згорткового шару, використовуваного для вилучення ознак, а також повнозв'язного шару, використовуваного для класифікації,

визначаються в процесі навчання. Вдосконалені мережеві структури CNN забезпечують економію пам'яті і складності обчислень і в той же час, забезпечують кращу продуктивність для додатків, де вхідні дані мають локальну кореляцію.

За рахунок накладення декількох різних шарів в CNN, створюються складні архітектури для вирішення задач класифікації. Найбільш поширені чотири типи шарів: згорткові шари (Convolution Layers), субдискретизуючі шари (Pooling Layers), нелінійні шари (Non-Linear Layers) і повнозв'язні шари (Fully Connected Layers).

Згортковий шар (Convolution Layers). Як випливає з назви, згортковий шар грає важливу роль в роботі CNN. Він утворює фундаментальну одиницю згорткової мережі всюди, де виконується велика частина обчислень. Параметри шару орієнтовані на використання ядер. Ці ядра зазвичай крихітні в просторовій розмірності, проте розгортаються у всьому вимірі глибини входу. Як тільки інформація потрапляє до згорткового шару, цей шар згортає кожен фільтр по просторовій розмірності даних, щоб надати двовимірну карту активації. Вихід нейронів, які підключені до локальних областей входу, можна перевірити через шар згортки шляхом обчислення скалярного добутку між їх вагами, а також площею, пов'язаною з вхідним обсягом. Нейрони, які складаються з ідентичної карти ознак, мають загальні ваги (поділ параметрів), тим самим зменшуючи складність мережі за рахунок збереження невеликої кількості параметрів [3]. Випрямляючий лінійний блок (більш відомий як ReLU) націлений на використання «поелементної» функції активації, такої як сигмоїдальна, на виході активації, зробленої попереднім шаром. Шари згортки можуть значно зменшити складність моделі за рахунок оптимізації її виходу. Однак вони оптимізовані за допомогою трьох гіперпараметрів: глибини, кроку і заповнення нулями. Заповнення нулями – це простий метод заповнення краю входу і ефективний метод забезпечення додаткового управління розмірністю вихідних обсягів. Щоб розрахувати

просторову розмірність вихідних даних згорткових шарів, зазвичай використовують таку формулу:

$$\frac{(V-R)+2Z}{S+1}, \tag{2.10}$$

де V представляє розмір вхідного масиву даних;

R – представляє розмір рефлекторного поля;

Z – являє собою величину набору нульового заповнення;

S – характеризує розмір кроку.

Процес згортки зображений на рисунку 2.5.

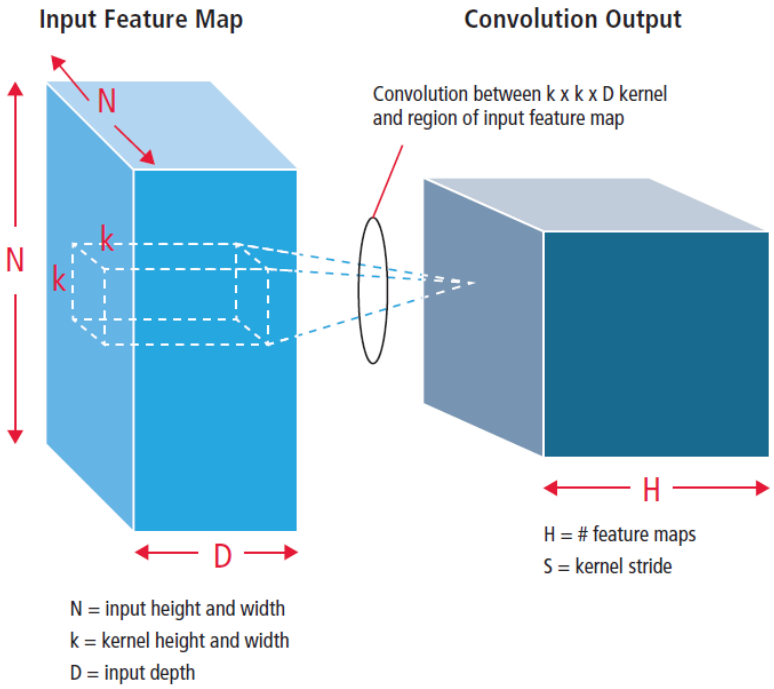

Рисунок 2.5 – Графічне зображення процесу згортки

Рисунок 2.6 демонструє процес 3D згортки, який також використовується в CNN.

Вхід величини H × H × W є згорнутим з p кількістю ядр, де розмір ядра k × k × W. Одне ядро, що згортає з вхідною картографічною

характеристикою, виробляє одну вихідну карту властивостей, а p ядро виробляє незалежно p карт. Кожне ядро переміщується, починаючи з верхнього лівого кута карти вхідних елементів у верхній правий кутовий елемент. Потім ядро переміщує один елемент вниз, займає ліве положення і рухається у напрямку з правого боку. Цей процес завершується, коли ядро досягає нижнього правого положення.

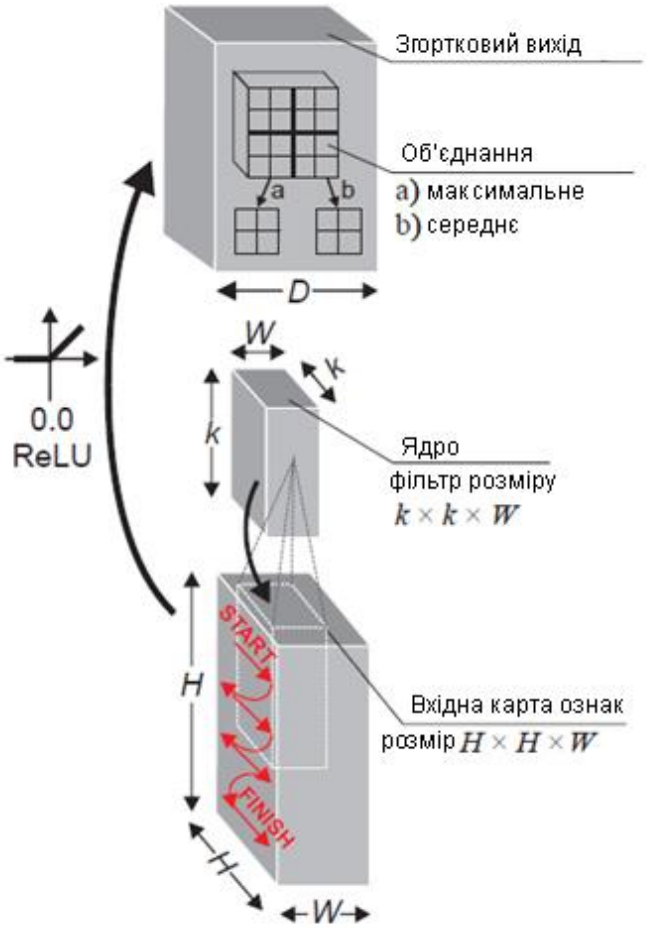

Рисунок 2.6 – Процес 3D згортки CNN

Наприклад, для випадку, коли ми маємо вхід $H \times H = 44$ і $k \times k$ 5, є 30 унікальних позицій зліва направо і 30 унікальних позицій від верху до низу, які ядро може зайняти. Кожна особливість у виводі згортки буде містити 30×30, тобто $(H - k + 1) \times (H - k + 1)$.

Наприклад $(44 - 5 + 1) \times (44 - 5 + 1)$ елементів. Для створення одного елемента з одного виходу потрібно зробити таку кількість операцій $k \times k \times W$.

З вищезазначених міркувань можна зробити висновок, що нова карта властивостей, як правило, генерується шляхом ковзання фільтра над входом і обчисленням точкового продукту (який подібний до операції згортки), за яким слідує нелінійна функція активації в моделі [5].

Наприклад, один згортковий шар складається з вхідної карти об'єкта, ядра і виходу згортки. Всі одиниці мають однакову вагу (фільтри) на кожній карті об'єктів.

Перевагою спільного використання вагових коефіцієнтів є зменшення кількості параметрів і можливість виявлення тієї самої функції незалежно від її розташування на входах. Максимальне об'єднання або середнє об'єднання використовуються для згортання виводу, щоб бути входом для наступного кроку згортки. Згорткові мережі навчаються за допомогою алгоритму зворотного поширення (backpropagation), він додатково включає операцію згортки з просторово перегорненими фільтрами. Окремий нейрон в вихідних даних може представляти градієнт, який може бути втрачено за глибиною, тому потрібно оновлювати тільки один набір ваг.

Нелінійний шар (Non-Linear Layer). Цей шар застосовує ненасичену функцію активації. Це збільшує нелінійні властивості функції вибору і всієї мережі, які бажані для багатошарових мереж, не зачіпаючи при цьому рецептивні поля згорткового шару. Зазвичай використовуються такі функції активації: сигмоїдальна (sigmoid), гіперболічний тангенс (tanh) і ReLU. У порівнянні з іншими функціями ReLU э найкращою, так як вона може навчати нейронні мережі в багато разів швидше. Крім того, для поліпшення продуктивності мережі в кінці останнього рівня використовується функція активації softmax.

ReLU можна описати рівнянням виду:

$$g(y) = \begin{cases} 0, y < 0 \\ y, y \geq 0, \end{cases} \quad (2.11)$$

де $g(y)$ – функція активації;

y – поточне значення.

ReLU реалізує функцію $y = max(x, 0)$, тому вхідні і вихідні розміри цього шару збігаються. Це збільшує нелінійні властивості функції прийняття рішень і всієї мережі, не впливаючи на рецептивні поля згорткового шару. У порівнянні з іншими нелінійними функціями, перевага ReLU полягає в тому, що мережа навчається у багато разів швидше. Функціональність ReLU показана на рисунку 2.7, а її передатна функція показана над стрілкою.

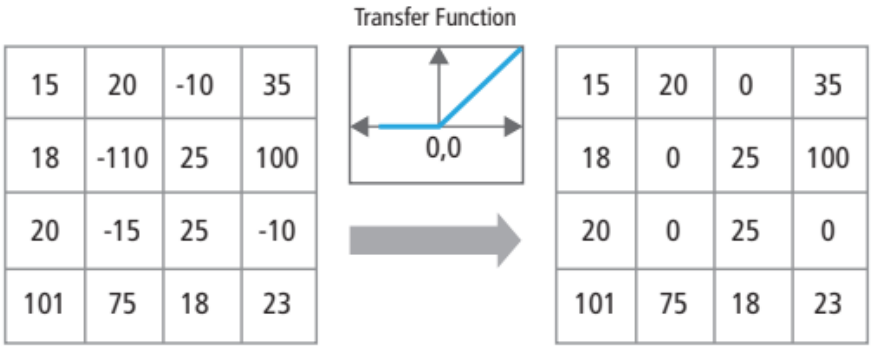

Рисунок 2.7 – Часткове представлення функції ReLU

Функція softmax може застосовуватися на останньому шарі згортки нейронної мережі (рисунок 2.8). Функція softmax може бути представлена наступним рівнянням:

$$a(c)_j = \frac{e^{c_j}}{\sum_{k=1}^{K} e^{c_k}}, \quad (2.12)$$

де $a(c)$ – функція активації;

c – поточне значення;

K – багатовимірний вектор;

k – індекс елементу вектору.

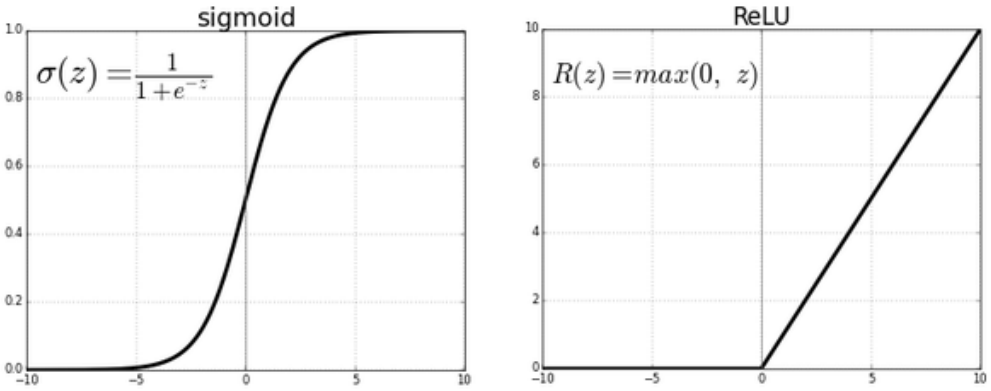

Рисунок 2.8 – Графічне представлення sigmoid та ReLu функцій активації

Субдискретизуючі шари (Pooling Layers). CNN містить не тільки згорткові шари, але і кілька субдискретизуючих шарів. Цей шар може бути розміщено одразу після згорткового шару. Це передбачає, що виходи згорткових шарів є входами для субдискретизуючих шарів мережі. Операції об'єднання скорочують розміри карт характеристик за рахунок використання деяких функцій для узагальнення субрегіонів, наприклад, взяття загального або максимального значення. Шари субдискретизації націлені на поступове зменшення розмірності даних і, отже, додаткове скорочення кількості параметрів, а також складність процедури моделі і, таким чином, контроль над питанням перенавчання. Існує загальний ряд операцій об'єднання – це максимальне об'єднання, середнє об'єднання, стохастичне об'єднання, спектральне об'єднання, об'єднання просторової піраміди, об'єднання по нормі L2 і багато масштабне об'єднання без порядку. На рисунку 2.9 показані операції максимального та середнього об'єднань.

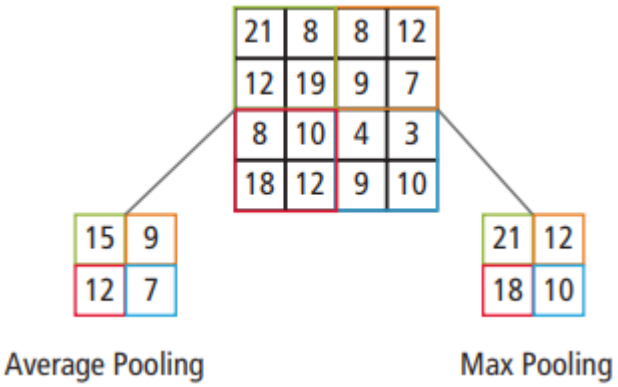

Рисунок 2.9 – Максимальне та середнє об’єднання

Кожна карта виводу може комбінувати згортку з кількома картами вводу. Загалом, можна записати:

$$x_j^L = f\left(\sum_{i \in M_i} x_j^{L-1} * k_{ij}^L + b_j^L\right), \tag{2.13}$$

де L – згортковий шар;

$L - 1$ – шар зменшення роздільної здатності карт;

x^{L-1} – вхідні особливості згорткового шару $L - 1$;

k_{ij} – карти ядра згорткового шару L;

b^L – адитивне зміщення згорткового шару L;

M_j – представляє вибір вхідних карт;

i – вхід;

j – вихід.

Загалом, вилучення об'єктів за допомогою CNN складається з декількох подібних кроків, і кожен крок складається з трьох каскадних шарів: шар згортки, шару активації та функції об'єднання. На рисунку 2.10 зображено приклад роботи повнозв’язного шару.

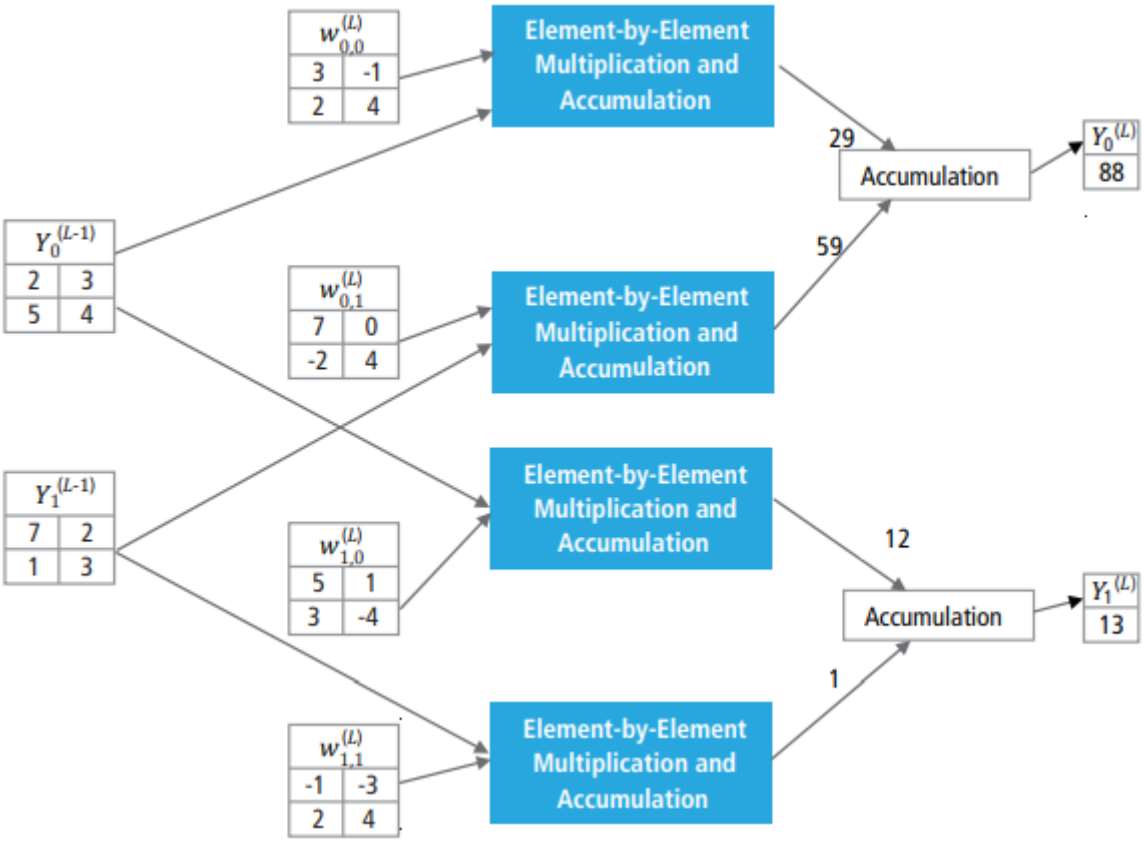

Рисунок 2.10 – Приклад роботи повнозв’язного шару

Повнозв’язний шар. Архітектура нейронної мережі завершується повнозв'язним шаром (рисунок 2.11). У цьому шарі нейрони на якомусь етапі мають зв'язки з усіма активаціями на попередньому шарі, як це зроблено в звичайних нейронних мережах багатошарового персептрона. Таким чином, їх активації будуть обчислюватися за допомогою матричної операції, за якою слідує упереджене зміщення.

Оскільки ми можемо витягувати високо рівневі характеристики вхідних зображень з вихідних даних шарів згортки і об'єднання, додавання повно зв’язного шару додатково є недорогим підходом для вивчення нелінійних комбінацій цих функцій. Хоча більшості функцій зі згорткових і об'єднуючих шарів може бути досить для завдання класифікації; проте комбінації цих функцій можуть бути навіть краще [4]. Нейрони в повно зв’язному шарі просторово не організовані, тому в ньому не може бути

згорткового шару. Повно зв’язний шар передає двовимірний результат на вихідний шар, де ми можемо використовувати функцію softmax або сигмоїду для передбачення мітки вхідного класу.

2.4 GoogleNet

Починаючи з LeNet-5, згорткові нейронні мережі (CNN) зазвичай мали стандартну структуру – за складеними згортковими шарами (необов'язково з наступною нормалізацією контрасту і maxpooling) слідують один або кілька повнозв’язаних шарів. Варіанти цього базового дизайну широко поширені в літературі по класифікації зображень і дали кращі на сьогоднішній день результати по MNIST, CIFAR і, в першу чергу, по завданню класифікації ImageNet. Для більших наборів даних, таких як Imagenet, недавня тенденція полягала в збільшенні кількості шарів [7] і розміру шарів при одночасному використанні dropout для вирішення проблеми перенавчання.

Мережа-в-мережі – це підхід, запропонований Lin [7], щоб збільшити репрезентативну потужність нейронних мереж. Стосовно до згорткових шарів, метод можна розглядати як додаткові 1×1 згортальні шари, за якими зазвичай слідує випрямлена лінійна активація (ReLu). Це дозволяє легко інтегрувати його в існуючі CNN. Цей підхід активно використовується в архітектурі GoogleNet. Однак в поточних умовах згортки 1×1 має подвійне призначення: що найбільш важливо, вони використовуються в основному як модулі зменшення розмірності для усунення обчислювальних вузьких місць, які в іншому випадку обмежили б розмір мережі. Це дозволяє не тільки збільшувати глибину, але і ширину мережі без значного зниження продуктивності.

В даний час провідним підходом до виявлення об'єктів є R-CNN (Regions CNN) мережі, запропоновані Girshick [8]. R-CNN розбиває загальну проблему виявлення на дві підзадачі: спочатку використовувати

низькорівневі сигнали, такі як узгодженість кольору і суперпікселі для потенційних пропозицій об'єктів без урахування категорій, а потім використовувати класифікатори CNN для визначення категорій об'єктів в цих місцях. Такий двоетапний підхід використовує точність сегментації прямокутника з низькорівневими репліками, а також потужні можливості класифікації сучасних CNN. В GoogleNet використовується аналогічний підхід, але дослідження показали поліпшення на обох етапах, таких як прогнозування з використанням декількох блоків і ансамблеві підходи для кращої категоризації пропозицій.

Найпростіший спосіб підвищити продуктивність глибинних нейронних мереж – це збільшити їх розмір. Це включає як збільшення глибини (кількості шарів) мережі, так і її ширини: кількості одиниць на кожному шарі. Це простий і безпечний спосіб навчання моделей більш високої якості, особливо з урахуванням наявності великої кількості помічених тренувальних даних. Однак це просте рішення має два основних недоліки.

Більший розмір зазвичай означає більшу кількість параметрів, що робить розширену мережу більш схильною до перенавчання, особливо якщо кількість помічених прикладів в навчальному наборі обмежена. Це може стати серйозним вузьким місцем, оскільки створення високоякісних навчальних наборів може бути складним і дорогим, особливо якщо необхідні досвідчені люди-оцінювачі, щоб розрізняти дрібнозернисті візуальні категорії, такі як в ImageNet, як показано на рисунку 2.11.

Ще один недолік рівномірно зростаючого розміру мережі – різко зростаюче використання обчислювальних ресурсів. Наприклад, в мережі глибинного зору, якщо два згортальних шари пов'язані між собою, рівномірне збільшення кількості їх фільтрів призводить до квадратичного збільшення обчислень. Якщо додана ємність використовується неефективно (наприклад, якщо більшість ваг в кінцевому підсумку виявляється близьким до нуля), то втрачається багато обчислень. Оскільки

на практиці обчислювальний бюджет завжди обмежений, ефективний розподіл обчислювальних ресурсів краще не виборчого збільшення розміру, навіть якщо основною метою є підвищення якості результатів.

Рисунок 2.11 – Сибірський хаскі та ескімоська собака

Фундаментальний спосіб вирішення обох проблем полягав би в остаточному переході від повністю зв'язаних архітектур до рідко зв'язаних архітектур, навіть всередині згорток. Крім імітації біологічних систем, це також буде мати перевагу у вигляді більш міцного теоретичного обґрунтування завдяки новаторської роботі Arora [9]. Їх основний результат полягає в тому, що якщо розподіл ймовірностей набору даних може бути представлено рівним, дуже розріджена глибинна нейронна мережа, то оптимальна топологія мережі може бути побудована шар за шаром шляхом аналізу статистики кореляції активацій останнього рівня і кластеризація нейронів з сильно корельованими виходами. Хоча строгий математичний доказ вимагає дуже строгих умов, той факт, що це твердження резонує з добре відомим принципом Хебба – нейрони, які спрацьовують разом, з'єднуються один з одним, – припускає, що основна ідея – може бути застосована на практиці навіть при менш суворих умовах.

З іншого боку, сучасні обчислювальні інфраструктури дуже неефективні, коли справа доходить до чисельних розрахунків неоднорідних, розріджених структур даних. Навіть якщо кількість арифметичних операцій зменшиться на 100, накладні витрати на пошук і промахи в кеші будуть настільки домінуючими, що перехід на розріджені матриці не окупиться. Також розрив ще більше збільшується за рахунок використання покращених, ретельно налаштованих числових бібліотек, які забезпечують надзвичайно швидке множення щільних матриць, використовуючи дрібні деталі базового апаратного забезпечення.

Крім того, неоднорідні розріджені моделі вимагають більш складної інженерної та обчислювальної інфраструктури. Більшість сучасних систем машинного навчання, орієнтованих на зір, використовують просторову розрідженість тільки за рахунок використання згорток. Однак згортки реалізовані як набори щільних з'єднань з латками на більш ранньому рівні. У згорткових мереж традиційно використовуються таблиці випадкових і розріджених з'єднань у вимірах функцій, щоб порушити симетрію і поліпшити навчання та згодом тенденція змінилася назад на повні зв'язки, щоб краще оптимізувати паралельні обчислення. Однаковість структури, велика кількість фільтрів і більший розмір пакета дозволяють використовувати ефективні обчислення.

Це піднімає питання, чи є будь-яка надія на наступний, проміжний крок – архітектуру, яка використовує додаткову розрідженість навіть на рівні фільтра, яку передбачає теорія, використовуючи обчислення на щільних матрицях. Багато літератури щодо обчислення розріджених матриць передбачає, що кластеризація розріджених матриць у відносно щільні під-матриці має тенденцію забезпечувати сучасну практичну продуктивність для множення розріджених матриць.

Архітектура Inception почалася з тематичного дослідження для оцінки гіпотетичного результату складного алгоритму побудови топології мережі, який намагається апроксимувати розріджену структуру, обіцяну

для мереж комп’ютерного зору і покриває гіпотетичний результат щільними, легко доступними складовими частинами. Вже після двох ітерацій по точному вибору топології був помічений невеликий виграш в порівнянні з еталонною архітектурою. Після подальшого налаштування швидкості навчання, гіперпараметрів і поліпшеної методології навчання було встановлено, що отримана в результаті початкова архітектура була особливо корисна в якості базової мережі для виявлення об'єктів. Цікаво, що хоча більшість початкових архітектурних рішень були поставлені під сумнів і ретельно протестовані, вони виявилися оптимальними, що найменше, локально.

Однак слід бути обережним: хоча запропонована архітектура стала успішною для комп'ютерного зору, все ще є сумніви, чи можна віднести її якість до принципів, які привели до її створення. Щоб переконатися у цьому, потрібний набагато більш ретельний аналіз і перевірка: наприклад, якби автоматизовані інструменти, засновані на принципах, описаних нижче, знайшли б аналогічну, але кращу топологію для мереж технічного зору. Найбільш переконливим доказом була б автоматизована система, яка створює мережеві топології, що показує аналогічні переваги в інших областях, використовуючи той самий алгоритм, але з абсолютно іншою глобальною архітектурою.

Основна ідея архітектури Inception заснована на з'ясуванні того, як оптимальна локальна розріджена структура в згортковій мережі може бути апроксимована і покрита легкодоступними щільними компонентами (рисунок 2.13). Все, що потрібно зробити – це знайти оптимальну локальну конструкцію і повторити її в просторі. Arora в [10] пропонує пошарову конструкцію, при якій слід аналізувати статистику кореляції останнього шару і кластеризувати її в групи одиниць з високою кореляцією. Ці кластери утворюють блоки наступного шару і пов'язані з блоками попереднього шару. Припускається, що кожна одиниця з попереднього шару відповідає деякій області вхідного зображення і ці

одиниці згруповані в банки фільтрів. У нижніх шарах (близьких до входу) корельовані одиниці будуть концентруватися в локальних регіонах. Це означає, що буде багато кластерів, зосереджених в одній області, і вони можуть бути покриті шаром з 1×1 згорток в наступному шарі. Однак можна також очікувати, що буде менша кількість більш рознесених в просторі кластерів, які можуть бути покриті згортками на більших ділянках, і буде зменшуватися кількість ділянок на все більш і більш великих областях. Щоб уникнути проблем з узгодженням патчів, поточні версії архітектури Inception обмежені розмірами фільтрів 1×1, 3×3 та 5×5, проте це рішення було засноване більше для зручності, а не за необхідності. Це також означає, що запропонована архітектура являє собою комбінацію всіх цих шарів з їх банками вихідних фільтрів, об'єднаними в один вихідний вектор, який утворює вхід наступного етапу. Крім того, оскільки операції об'єднання були важливі для успіху в сучасних згорткових мережах, це передбачає, що додавання альтернативного шляху паралельного об'єднання на кожному такому етапі також має мати додатковий позитивний ефект [11].

Оскільки ці «модулі Inception» накладаються один на одного, їх статистика вихідної кореляції обов'язково буде змінюватися, оскільки характеристики вищої абстракції захоплюються більш високими шарами (рисунок 2.12). Очікується, що їх просторова концентрація буде зменшуватися, тож співвідношення 3×3 та 5×5 згорток повинні збільшуватися в міру переходу до більш високих шарів.

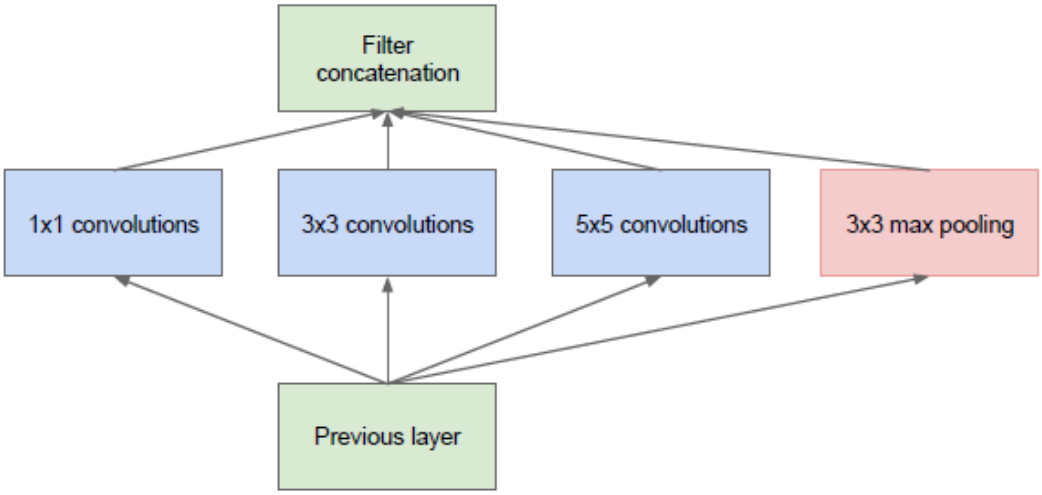

Рисунок 2.12 – Модуль Inception, проста версія

Велика проблема з вищезгаданими модулями полягає в тому, що навіть невелика кількість згорток в 5×5 може бути надмірно дорогою поверх згорткового шару з великою кількістю фільтрів. Ця проблема стає ще більш помітною, коли додаються блоки об'єднання, кількість їх вихідних фільтрів дорівнює кількості фільтрів на попередньому етапі. Об'єднання вихідних даних шару об'єднання з виходами згортальних шарів призведе до неминучого збільшення кількості вихідних даних від етапу до етапу. Навіть незважаючи на те, що ця архітектура може охоплювати оптимальну розріджену структуру, вона буде робити це дуже неефективно, що призведе до обчислювального вибуху за кілька етапів.

Це призводить до другої ідеї запропонованої архітектури – розумного застосування зменшення розмірності і проекції всюди, де в іншому випадку занадто сильно збільшуються обчислювальні вимоги. Це засновано на успішності вставок: навіть низько-розмірні вставки можуть містити багато інформації про відносно великі фрагменти зображення. Однак вставки представляють інформацію в щільній стислій формі, а стислу інформацію складніше змоделювати [12]. Але бажано, щоб уявлення було розрідженим в більшості місць, тому ми стискаємо сигнали тільки тоді, коли вони повинні бути агреговані в масі. Тобто 1×1 згортки

використовуються для обчислення скорочень перед дорогими 3×3 та 5×5 згортками. Крім використання в якості редукторів, вони також включають використання ReLu, що робить їх подвійними. Остаточний результат показаний на рисунку 2.13.

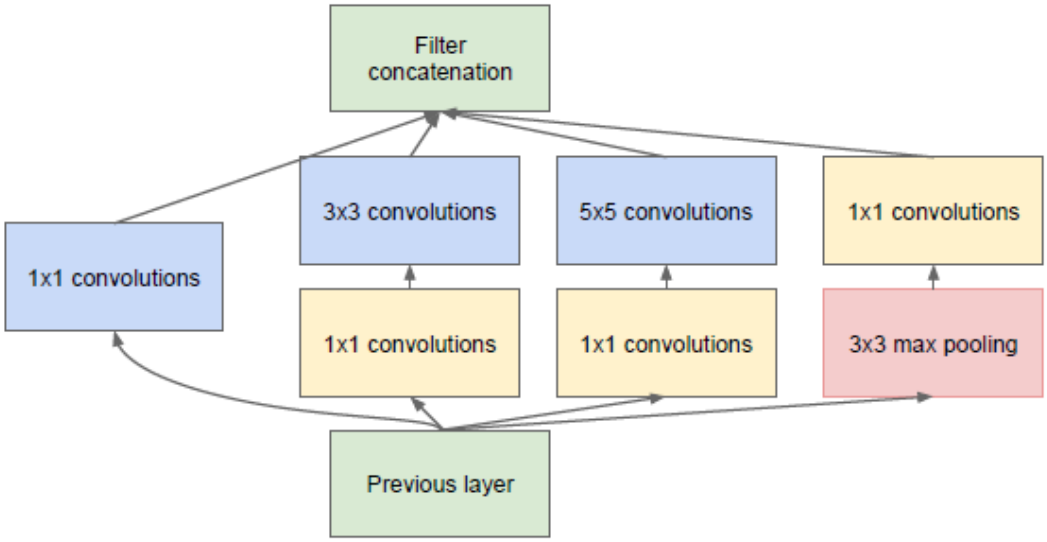

Рисунок 2.13 – Модуль Inception зі зменшенням розмірів

Загалом, Inception мережа – це мережа, що складається з модулів вищевказаного типу, накладених один на одного, з випадковими шарами максимального об'єднання з кроком 2, щоб вдвічі зменшити розширення сітки. З технічних причин (ефективність пам'яті під час навчання) здавалося вигідним починати використовувати модулі Inception тільки на більш високих шарах, зберігаючи при цьому нижні шари традиційним згортковим способом. Це не є строго необхідним, просто через деяку неефективність інфраструктури в поточної реалізації [13].

Одним з основних корисних аспектів цієї архітектури є те, що вона дозволяє значно збільшити кількість блоків на кожному етапі без неконтрольованого збільшення обчислювальної складності. Повсюдне використання зменшення розмірності дозволяє екранувати велику кількість вхідних фільтрів останнього етапу на наступний шар, спочатку

зменшуючи їх розмір, а потім згортаючи їх з великим розміром фрагмента. Ще один практично корисний аспект цієї архітектури полягає в тому, що він узгоджується з інтуїцією, що візуальна інформація повинна оброблятися в різних масштабах, а потім агрегуватися, щоб на наступному етапі можна було одночасно абстрагувати функції з різних масштабів. Архітектура мережі наведена в таблиці 2.1.

Поліпшене використання обчислювальних ресурсів дозволяє збільшувати як ширину кожного етапу, так і кількість етапів без виникнення обчислювальних труднощів [14]. Ще один спосіб використовувати архітектуру Inception – створити трохи гіршу, але дешевшу в обчисленні версію. Було виявлено, що всі включені «ручки і важелі» дозволяють здійснювати контрольоване балансування обчислювальних ресурсів, що може привести до створення мереж, які в 2-3 рази швидші, ніж мережі з аналогічною продуктивністю з не Inception архітектурою, проте на даному етапі це вимагає ретельного ручного проектування.

Всі згортки, в тому числі всередині модулів Inception, використовують ReLu. Розмір рецептивного поля в мережі, яке приймає канали кольору RGB, становить 224×224. $\#3 \times 3$ зменшення та $\#5 \times 5$ зменшення позначають кількість фільтрів 1×1, що зменшує шар, що використовується, до 3×3 та 5×5 згорток. Після вбудованого max-pooling в стовпці pool proj можна побачити кількість фільтрів в шарі проекції. Всі ці шари зменшення / проекції також використовують ReLu [15].

Таблиця 2.1 – Архітектура мережі на основі Inception

Тип	Вхід	Вихід	Глиб.	#1 × 1	#3 × 3 зменш.	# 3 × 3	#5 × 5 зменш.	# 5 × 5	Poll proj	Парам.	OPS
convolution	$7 \times 7/2$	$112 \times 112 \times 64$	1							2.7K	34M
max pool	$3 \times 3/2$	$56 \times 56 \times 64$	0								
convolution	$3 \times 3/1$	$56 \times 56 \times 192$	2		64	192				112K	360M
max pool	$3 \times 3/2$	$28 \times 28 \times 192$	0								
inception (3a)		$28 \times 28 \times 256$	2	64	96	128	16	32	32	159K	128M
inception (3b)		$28 \times 28 \times 480$	2	128	128	192	32	96	64	380K	304M
max pool	$3 \times 3/2$	$14 \times 14 \times 480$	0								
inception (4a)		$14 \times 14 \times 512$	2	192	96	208	16	48	64	364K	73M
inception (4b)		$14 \times 14 \times 512$	2	160	112	224	24	64	64	437K	88M
inception (4c)		$14 \times 14 \times 512$	2	128	128	256	24	64	64	463K	100M
inception (4d)		$14 \times 14 \times 528$	2	112	144	288	32	64	64	580K	119M
inception (4e)		$14 \times 14 \times 832$	2	256	160	320	32	128	128	840K	170M
max pool	$3 \times 3/2$	$7 \times 7 \times 832$	0								
inception (5a)		$7 \times 7 \times 832$	2	256	160	320	32	128	128	1072K	54M
inception (5b)		$7 \times 7 \times 1024$	2	384	192	384	48	128	128	1388K	71M
avg pool	$7 \times 7/1$	$1 \times 1 \times 1024$	0								
dropout (40%)	$7 \times 7/1$	$1 \times 1 \times 1024$	0								
linear		$1 \times 1 \times 1000$	1							1000K	1M
softmax		$1 \times 1 \times 1000$	0								

Мережа була спроектована з урахуванням обчислювальної ефективності та практичності, тому логічний висновок може виконуватися на окремих пристроях, включаючи навіть ті, які мають обмежені обчислювальні ресурси, особливо з малим об'ємом пам'яті. Глибина мережі становить 22 рівня при підрахунку тільки шарів з параметрами (або 27 рівнів, якщо ми також враховуємо пули) [16]. Загальна кількість рівнів (незалежних будівельних блоків), які використовуються для побудови мережі, становить близько 100. Однак ця кількість залежить від використовуваної системи інфраструктури машинного навчання. Використання середнього пулу перед класифікатором засноване на [7], хоча ця реалізація відрізняється тим, що використовується додатковий лінійний шар. Це дозволяє легко адаптувати і налаштовувати мережі для інших наборів [17].

З огляду на відносно велику глибину мережі, проблемою була можливість ефективно поширювати градієнти назад по всім шарам. Одне цікаве відкриття полягає в тому, що висока продуктивність щодо дрібніших мереж в цьому завданні передбачає, що функції, які створюються шарами в середині мережі, повинні бути дуже різними.

Точна структура додаткової мережі на стороні, включаючи допоміжний класифікатор, виглядає наступним чином:

– середній рівень об'єднання з розміром фільтра 5×5 і кроком 3, в результаті чого на виході $4 \times 4 \times 512$ для (4a) і $4 \times 4 \times 528$ для (4d) стадії;

– згорток 1×1 із 128 фільтрами для зменшення розмірів та ReLu;

– повнозв’язні шари с 1024 частинами та ReLu;

– шар dropout з відкиданням виходів 70%;

– лінійний шар з втрати softmax в якості класифікатора.

3 РОЗРОБКА СИСТЕМИ ТА ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

3.1 Навчання нейронної мережі

Для навчання та тестування мережі використовувався набір даних, зібраний вручну. Набір складається з 25 000 зображень та 400 категорій. Зображення були зібрані з Інтернету: із загальнодоступних наборів даних, та зібраних вручну із платформи Instagram. Приклад рисунків з набору даних представлений на зображені на рисунку 3.1.

Рисунок 3.1 – Приклад зображень з набору даних

Набір складається із зображень зі змінною роздільною здатністю, в той час, як система вимагає постійної вхідної розмірності. Тому розмір зображень було зменшено до фіксованої роздільної здатності 256 × 256. Для прямокутного зображення спочатку був змінений масштаб зображення так, щоб його коротша сторона мала довжину 256, а потім вирізали центральний фрагмент 256 × 256 з отриманого зображення. Далі

зображення не були додатково обробленими будь-яким іншим способом. Отже, навчання мережі проводилося на необроблених значеннях RGB пікселів.

Стандартний спосіб змоделювати вихід f нейрона як функцію його входу x – це використовувати $f(x) = \tanh(x)$ або $f(x) = (1 + e - x) - 1$. З точки зору часу навчання з градієнтним спуском, ці насичують нелінійності набагато повільніше, ніж нелінійність без насичення $f(x) = max\,(0, x)$. Дотримуючись Наїр і Хінтон [2], ми називаємо нейрони з такою нелінійністю випрямленими лінійними одиницями (ReLU). Глибинні CNN з ReLU навчаються в кілька разів швидше, ніж їх еквіваленти з модулями tanh. Це показано на рисунку 3.2, де показано кількість ітерацій, необхідних для досягнення 25% помилки навчання в наборі даних CIFAR-10 для конкретної чотирьох-рівневої згорткової мережі. Цей графік показує, що в навчанні даної нейронної мережі не слід було б експериментувати з використанням традиційних моделей.

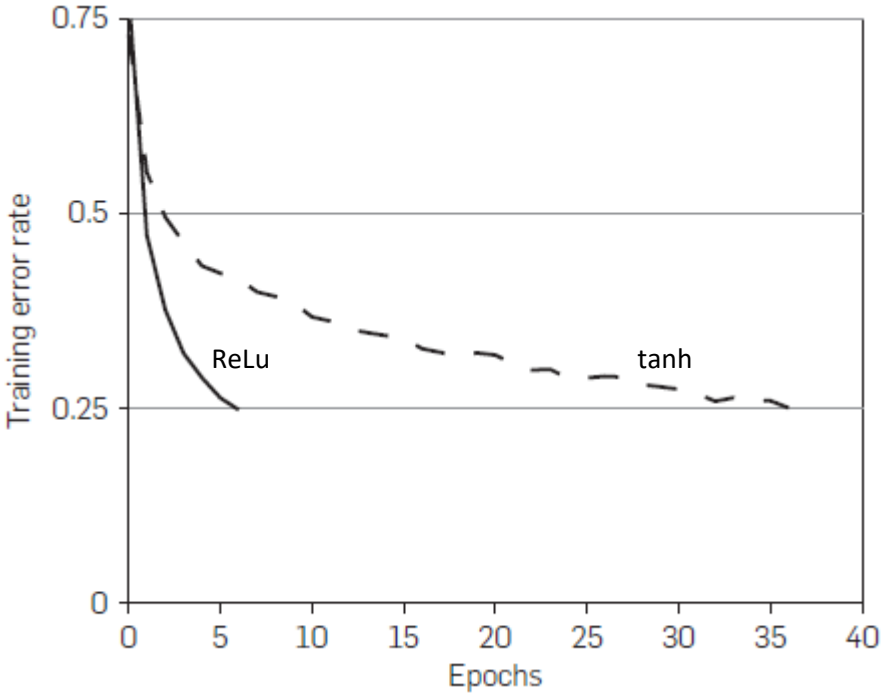

Рисунок 3.2 – Графік залежності часу навчання від функції активації, пунктир – tanh, суцільна – ReLU

Це не перший випадок, де розглядаються альтернативи традиційним моделям нейронів в CNN. Більш швидке навчання дуже сильно впливає на продуктивність великих моделей, навчених на великих наборах даних.

У ReLU є бажана властивість: вона не вимагає нормалізації входу, щоб запобігти його насичення. Якщо хоча б деякі навчальні приклади дають позитивний результат для ReLU, навчання відбуватиметься в цьому нейроні [18]. Однак наступна схема локальної нормалізації допомагає узагальненню. Позначаючи через $a_{x,y}^{i}$ активність нейрона, обчислену шляхом застосування ядра i в позиції (x, y) і подальшого застосування нелінійності ReLU, нормалізована з відповіді на активність $b_{x,y}^{i}$ задається виразом:

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + a \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^{j})^2 \right)^{\beta}, \tag{3.1}$$

де сума проходить по n «сусіднім» картам ядер в тій самій просторовій позиції;

N – загальна кількість ядер в шарі.

Порядок карт ядра, звичайно, довільний і визначається до початку навчання. Цей вид нормалізації відповіді реалізує форму бічного гальмування, натхненного типом, виявленим в реальних нейронах, створюючи конкуренцію за велику активність серед виходів нейронів, обчислених з використанням різних ядер [19]. Константи k, n, α і β є гіперпараметрами, значення яких визначаються з використанням набору для перевірки. В процесі навчання використовувались $k = 2, n = 5$, $\alpha = 10 - 4$ і $\beta = 0{,}75$. Дана нормалізація застосовувалася після ReLU в певних шарах.

Найпростіший і найпоширеніший метод зменшення перенавчання даних зображення – це штучне збільшення набору даних за допомогою

перетворень зі збереженням міток. Використовуються дві різні форми збільшення даних, обидві з яких дозволяють створювати перетворені зображення з вихідних зображень з дуже невеликим обсягом обчислень, тому перетворені зображення не потрібно зберігати на диску. В цій реалізації перетворені зображення генеруються в коді Python на CPU, в той час як GPU навчається на попередньому пакеті зображень. Таким чином, ці схеми збільшення даних фактично вільні від обчислень [20].

Перша форма збільшення даних складається з генерації перетворень зображень і горизонтальних відображень. Це робиться шляхом витягування випадкових ділянок розміром 224 × 224 (і їх горизонтальні відображення) з зображень розміром 256 × 256 і навчання мережі на цих ділянках [21]. Це збільшує розмір навчального набору в 2048 раз, хоча отримані приклади для навчання, звичайно, дуже взаємозалежні. Без цієї схеми мережа страждає від значного перенавчання, що змусило б використовувати мережі набагато меншого розміру. Під час тестування мережа робить прогноз, витягуючи п'ять фрагментів 224 × 224 (чотири кутових фрагмента і центральний фрагмент), а також їх горизонтальні відображення (отже, всього 10 фрагментів).

Друга форма збільшення даних полягає в зміні інтенсивності каналів RGB в навчальних зображеннях. Зокрема, в даній роботі виконується PCA для набору значень пікселів RGB у всьому навчальному наборі. До кожного зображення додаються кратні числа знайдених головних компонентів з величинами, пропорційними відповідним власним значенням, помноженим на випадкову величину, взяту з гауссіану із середнім 0 і стандартним відхиленням 0,1. Тому до кожного пікселя зображення RGB $I_{xy} = [I_{xy}^R + I_{xy}^G + I_{xy}^B]^T$ ми додаємо наступну величину:

$$[p_1, p_2, p_3][a_1\lambda_1, a_2\lambda_2, a_3\lambda_3]^T, \tag{3.2}$$

де p_i та λ_i – і-й власний вектор і власне значення коваріаційної матриці значень 3×3 пікселів RGB, відповідно;

a_i – вищезгадана випадкова величина.

Кожна a_i малюється тільки один раз для всіх пікселів конкретного тренувального зображення, поки це зображення не буде знову використано для тренування, після чого воно буде повторно намальовано. Ця схема приблизно відображає важливу властивість природних зображень, а саме те, що ідентичність об'єкта інваріантна до змін інтенсивності і кольору освітлення. Ця схема знижує частоту помилок першого порядку більш ніж на 1%.

Комбінування прогнозів багатьох різних моделей – дуже успішний спосіб зменшити помилки тестування, але, схоже, це занадто дорого для великих нейронних мереж, навчання яких вже займає кілька днів. Однак існує дуже ефективна версія комбінації моделей, яка коштує приблизно в два рази більше часу від часу навчання. Нещодавно представлений метод, названий «dropout» [1], полягає в обнуленні вихідного сигналу кожного прихованого нейрона з імовірністю 0,5. Нейрони, які «випадають» таким чином, не беруть участі в прямому проході і не беруть участь в зворотному поширенні. Таким чином, кожен раз нейронна мережа створює зразки іншої архітектури, але всі ці архітектури мають загальні ваги. Цей метод знижує складність адаптації нейронів, оскільки нейрон не може покладатися на присутність певних інших нейронів. Тому він змушений вивчати більш надійні функції, які корисні в поєднанні з багатьма різними випадковими підмножинами інших нейронів. Під час тестування використовувався «dropout» зі значенням 04, що є розумним наближенням до взяття середнього геометричного прогнозованих розподілів, створюваних мережами з експоненціально-великим числом «dropout».

В даній мережі використовувався «dropout» в перших двох повнозв'язних шарах. Без «dropout» мережа демонструє значне

перенавчання. «Dropout» приблизно вдвічі збільшує кількість ітерацій, необхідних для сходження.

Модель навчалась з використанням стохастичного градієнтного спуску з розміром партії 128 прикладів, імпульсом 0,9 і спаданням ваги 0,0005. Було виявлено, що це невелике зниження ваги важливе для навчання моделі. Іншими словами, зменшення ваги тут не просто є регулятором, воно зменшує помилку навчання моделі. Правило поновлення ваги w наступне:

$$\begin{aligned} v_{i+1} &:= 0.9 \cdot \epsilon \cdot w_i - \epsilon \cdot \langle \frac{\partial L}{\partial w} |_{w_i} \rangle_{D_i}, \\ w_{i+1} &:= w_i + v_{i+1}, \end{aligned} \tag{3.3}$$

де i – індекс ітерації;

v – змінна імпульсу;

ϵ – швидкість навчання;

$\langle \frac{\partial L}{\partial w} |_{w_i} \rangle_{D_i}$ – середнє по i-му пакету D_i похідною по w, обчисленої в w_i.

Ваги ініціалізуються на кожному шарі за розподілом Гауса з нульовим середнім зі стандартним відхиленням 0,01. Зміщення нейронів ініціалізується у другому, четвертому і п'ятому згортальних шарах, а також в повністю пов'язаних прихованих шарах з константою 1. Ця ініціалізація прискорює ранні стадії навчання, забезпечуючи ReLU з позитивними вхідними даними. Зміщення нейронів в інших шарах ініціалізується константою 0.

Для навчання використовувалась однакова швидкість для всіх рівнів, яка налаштовувалась вручну протягом усього навчання. Евристика полягала в тому, щоб розділити швидкість навчання на 10, коли частота помилок перевірки перестає поліпшуватися з поточною швидкістю навчання. Швидкість навчання була ініціалізована на рівній 0,01 і знижена в три рази перед завершенням.

Мережа навчалась 30 ітерацій на навчальному наборі з 25 тисяч зображень, що зайняло 6 годин на графічному процесорі NVIDIA RTX 2070 SUPER 8 GB. На кожній ітерації зберігалася модель та оцінка точності. Результати наведені нижче в таблиці 3.1.

Таблиця 3.1 – Точності навчання мережі на різній кількості ітерацій

Кількість ітерацій	Точність, %
5 000	90.07
4 000	88.53
3 000	86.22
2 000	82.91
1 000	75.17

На рисунку 3.3 представлено графік залежності точності класифікації від кількості ітерацій.

На першій ітерацій точність сягала 33.17% та поступово зростала з ростом кількості ітерацій. Для вибору оптимальної кількості ітерацій було проведено навчання мережі з кількістю ітерацій 30 – збільшення точності відбувалося на дуже малі значення. Свого піку мережа досягла на 9 ітерації.

Також може бути створено декілька варіантів моделей. Їх практичні відмінності полягають у різниці кількості параметрів, що напряму впливає на швидкість навчання та кінцеву точність. Варіант моделі залежить від середовища, де вона буде використовуватися. Наприклад, модель, що працює в центрі обробки даних, може мати багато параметрів і через це вимагає великої кількості операцій в секунду. Тоді як модель, що працює на мобільному телефоні, повинна мати невелику кількість параметрів, адже середовище має дуже обмежену кількість ресурсів.

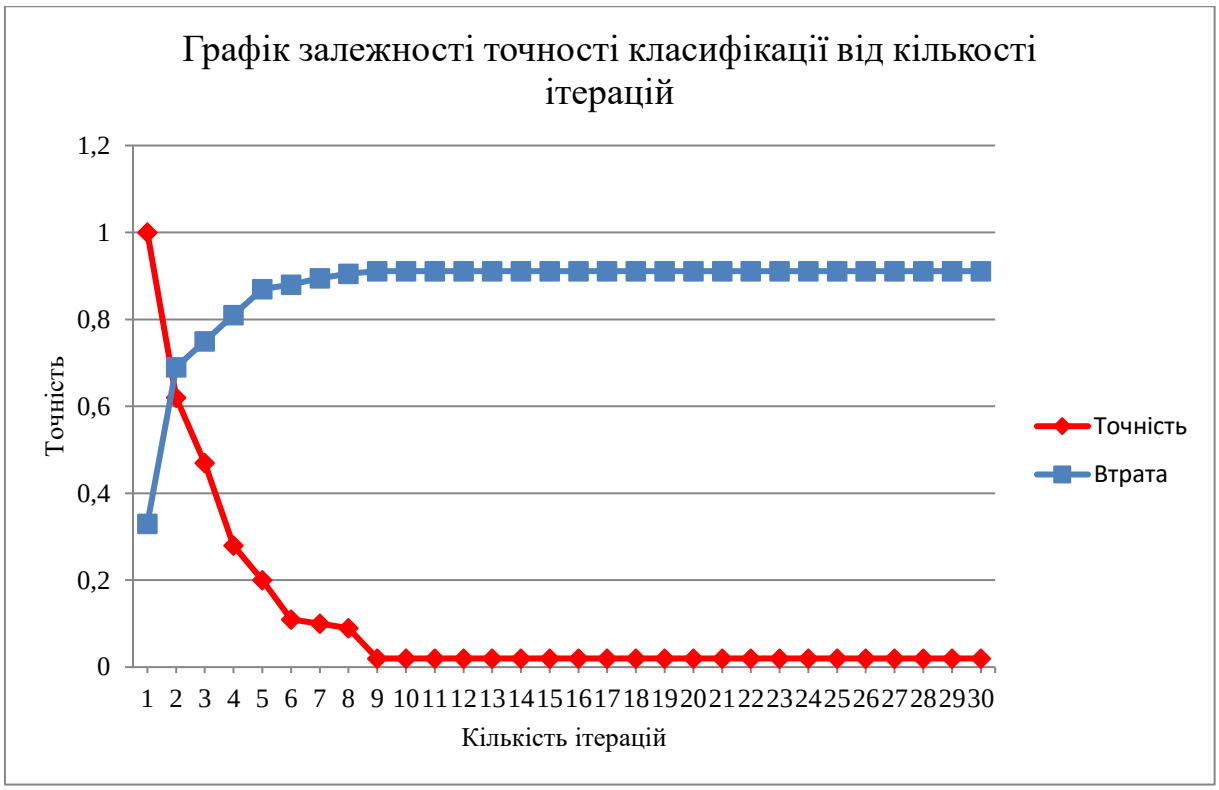

Рисунок 3.3 – Залежність точності класифікації та коефіцієнту втрати помилки від кількості ітерацій

Для дослідження залежності точності мережі від попередньої обробки зображень були застосовані 4 різні методи.

– вхідне зображення у кольоровому просторі RGB (рисунок 3.4);

– конвертація вхідного RGB-зображення в градації сірого (рисунок 3.5);

– конвертація вхідного RGB-зображення у кольоровий простір HSV (рисунку 3.6);

Рисунок 3.4 – Вхідне зображення у кольоровому просторі RGB

Рисунок 3.5 – Вхідне RGB-зображення в градаціях сірого

Рисунок 3.6 – Вхідне RGB-зображення у кольоровому просторі HSV

– конвертація вхідного RGB-зображення у кольоровий простір HSV та градації сірого (рисунок 3.7). Для проведення такої обробки зображення було конвертовано у кольоровий простір HSV, з якого був обраний канал value (яскравість).

Рисунок 3.7 – Вхідне RGB-зображення у кольоровому просторі HSV та градаціях сірого

Порівняння точності класифікації від вибору попередньої обробки зображень представлено на рисунку 3.8:

Рисунок 3.8 – Порівняння точності класифікації від вибору попередньої обробки зображень

Як видно з рисунку 3.8, найкращі результати були отримані для зображень у кольоровому просторі RGB. При такій обробці мережі на вхід подається більша кількість інформації, таким чином вона може вивчати кілька функцій для класифікації зображень. Також важливо відзначити, що тренування мережі на зображеннях, які були конвертовані в градації сірого, дозволяють отримати високу точність на тренувальному наборі, проте точність на тестувальному наборі є найнижчою серед усіх досліджень. Дану точність можна пояснити явищем перенавчання. Оскільки зображення у градаціях сірого втрачають важливі ознак, мережа втрачає здатність до узагальнення.

3.2 Розробка Web-сервісу

WEB-сервіс був розроблений на платформі Node.js. Node.js – це платформа, побудована на середовищі виконання JavaScript для легкого створення швидких і масштабованих мережевих програм. Node.js використовує модуль вводу-виводу, керовану подіями, що робить його легким і ефективним.

Node.js є open-source продуктом та крос-платформним середовищем для розробки серверних і мережевих додатків. Програми Node.js написані на JavaScript, і можуть бути запущені в середовищі виконання Node.js на ОС X, Microsoft Windows і Linux.

Node.js також надає багату бібліотеку різних модулів JavaScript, що значною мірою спрощує розробку веб-додатків.

Також платформа Node.js була обрана для розробки сервісу через такі переваги:

– асинхронне та Event Driven управління – всі API бібліотеки Node.js є асинхронними, тобто неблокуючими. Це означає, що сервер на основі Node.js ніколи не чекає на повернення даних API. Сервер переходить до наступного API після виклику, а механізм сповіщень Events Node.js допомагає серверу отримувати відповідь від попереднього виклику API;

– швидкість роботи – бібліотека Node.js Google Chrome Engine V8, дуже швидко виконує код;

– однопоточність, але масштабованість – Node.js використовує одну потокову модель з циклом подій. Механізм подій допомагає серверу реагувати неблокуючим способом і робить сервер дуже масштабованим, на відміну від традиційних серверів, які створюють обмежені потоки для обробки запитів. Node.js використовує одну потокову програму, і одна й та сама програма може надавати послуги набагато більшій кількості запитів, ніж традиційні сервери, такі як HTTP-сервер Apache.

Далі була розроблена архітектура WEB-сервісу. В даній архітектурі було використано багато хмарних сервісів, оскільки вся система буде працювати в середовищі AWS.

Нижче на рисунку 3.9 представлена архітектура WEB-сервісу.

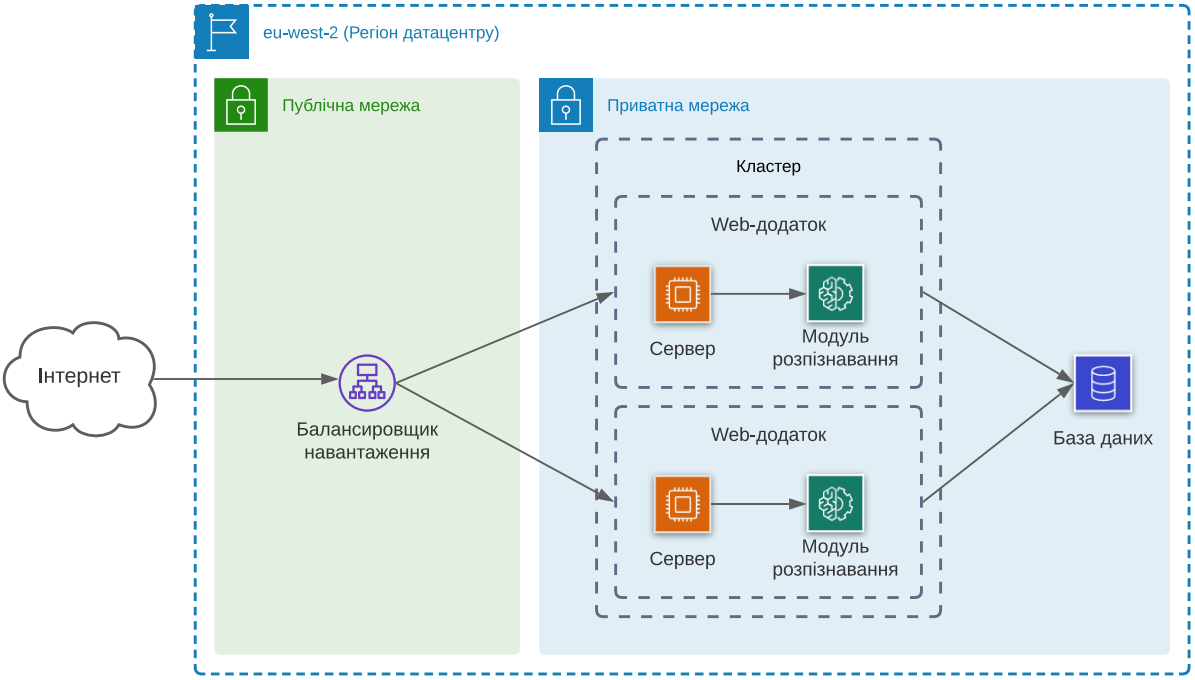

Рисунок 3.9 – Архітектура Web-сервісу

Запити від користувачів надходять з Інтернету. В свою чергу балансировщик навантаження розподіляє запити по різним серверам. Кількість одночасно працюючих серверів – необмежена. Кожен сервер знаходиться в своїй зоні доступності і у випадку відмови якогось серверу, будь-який інший може прийняти додаткове навантаження.

Кожен сервер має два модулі:

– сервер, з яким безпосередньо працюють користувачі;

– модуль розпізнавання, з яким безпосередньо працює сервер.

Модуль машинного зору вимагає обробки великої кількості зображень. Для запуску CNN у вбудованій системі, яка підтримує обробку зображень, вона повинна відповідати наступним вимогам:

– доступність високої обчислювальної продуктивності: для типової реалізації CNN необхідні мільярди MAC в секунду;

– велика пропускна здатність завантаження або збереження. В разі повно зв’язного шару, який використовується для цілей класифікації, кожен коефіцієнт використовується при множенні тільки один раз. Таким чином, вимоги до пропускної здатності більше, ніж кількість MAC, які виконуються процесором;

– фіксована потужність: система повинна споживати менше енергії. Для вирішення цієї проблеми потрібна реалізація з фіксованою точкою, яка вимагає дотримання вимог до продуктивності з використанням мінімально можливого кінцевого числа бітів;

– гнучкість: повинна бути можливість легко модернізувати існуючу архітектуру до нової, більш ефективної архітектури.

Оскільки обчислювальні ресурси завжди є обмеженням у системах, якщо варіант використання допускає невелике зниження продуктивності, корисно мати алгоритм, який може забезпечити величезну економію обчислювальної складності за рахунок невеликого контрольованого зниження продуктивності. Робота над алгоритмом досягнення компромісу між складністю і продуктивністю, як пояснювалося в попередньому розділі, має велике значення для впровадження CNN у системах.

Для зберігання та обробки даних використовується СУБД MySQL. MySQL – компактний багато потоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. Дана СУБД вважається гарним рішенням для малих і середніх застосувань та має такі переваги:

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із БД;

– кількість записів у таблицях може досягати 50 млн;

– висока швидкість виконання команд.

ВИСНОВКИ

В атестаційній роботі представлені результати, котрі відповідають поставленій меті, і є вирішенням актуального завдання виявлення та розпізнавання зображення за допомогою згорткової нейронної мережі, котра забезпечує високу продуктивність та точність.

Проведені дослідження дозволяють зробити наступні висновки:

– у ході проведення досліджень було виявлено три основі проблеми розпізнавання зображень за допомогою нейронних мереж: вибір архітектури мережі, обчислювальна потужність комп’ютера, формування тренувальної вибірки та запропоновані шляхи їх вирішення;

– в рамках атестаційної роботи була розроблена архітектура загорткової нейронної мережі. Такі переваги обраної мережі, як швидкість навчання, можливість розпізнавання зображення з різних ракурсів та різного масштабу та можливість навчати нейронну мережу на графічних процесорах, дозволяють вирішити поставлену задачу;

– проведено тестування різних конфігурацій мережі та обрано конфігурацію, що дозволяє отримати найкращу точність;

– проведено навчання мережі з різними методами первинної обробки зображення та визначено метод, що дозволяє досягти найбільшої точності;

– розроблена архітектура WEB-сервісу дозволяє без будь-яких ускладнень масштабувати програмне забезпечення та швидко налаштовувати середовище виконання;

– програмне забезпечення розроблене за допомогою найсучасніших хмарних технологій AWS, що дозволяє контролювати всі етапи життєвого циклу системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 30 p.;

2. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition / C. Szegedy et. al. New York, NY: Columbia University Press, 2015. P. 1–9;

3. Improving neural networks by preventing co-adaptation of feature detectors / G. E. Hinton, et. al. arXiv preprint arXiv:1207.0580, 2012 101 p.;

4. Striving for simplicity: The all convolutional net / J. T. Springenberg, et. al. arXiv preprint arXiv:1412.6806, 2014. 210 p.;

5. A. Albelwi, A. Mahmood. A Framework for Designing the Architectures of Deep Convolutional Neural Networks, Entropy, vol. 19, no. 6, 2017. 242 p.;

6. Karpathy A., Fei-Fei L. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Red Hook, NY: Curran, 2016. P. 3128–3137;

7. Min Lin, Qiang Chen, Shuicheng Yan. Network in network. CoRR: 1312.4400, 2013. 57 p.;

8. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition / Girshick J et. al., CVPR: IEEE Conference, 2014. 541 p.;

9. Provable bounds for learning some deep representations / A. Sanjeev et. al. CoRR:1310.6343, 2013. 256 p.;

10. Overfeat: Integrated recognition, localization and detection using convolutional networks / P. Sermanet et. al. CoRR: 1312.6229, 2013. 189 p.;

11. I.Namatevs. Deep Convolutional Neural Networks: Structur. Feature Extraction and Training, vol. 20, 2017. P. 40–47;

12. W. Rawat, Z.Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. CoRR: 1310.6343, 2017. 98 p.;

13. Nielsen M. Deep Learning. Neural Networks and Deep Learning online book. CoRR: 1310.6343, 2017. 78 p.;

14. Learning activation functions to improve deep neural networks / G. H. Forest et. al. CoRR: 1310.6343, 2014. 171 p.;

15. Крижевский A. ImageNet Classification with Deep Convolutional Neural Networks. CoRR: 1312.4659, 2013. P. 57–59;

16. Zeiler M., Fergus P. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. CoRR: 1312.4659, 2013. 62 p.;

17. А.А., Милешин. Алгоритмы распознавания рукописных подписей на основе нейронных сетей. Фундаментальные исследования. № 11-9. 2014. P. 1906–1910;

18. Abdallah A., Abou E., Lynn A., A New Face Detection Technique using 2D DCT and Self Organizing Feature Map in Proc. of World Academy of Science Engineering and Technology. vol. 21, 2007. P. 15-19;

19. Törmä M., Kohonen. Self-organizing feature mapin pattern recognition, Institute of Photogrammetry and Remote Sensing Helsinki University of Technology, 2005. 14 p.;

20. Schmidt M. Optimization Methods for L1-Regularization, USA, 2009. 120 p.;

21. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / N. Srivastava, Journal of Machine Learning Research 15, 2014. P. 1929-1958.