НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах
рукопису» УДК
044.77

«До захисту допущено»

Завідувач кафедри СПСКС

_____

В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Спеціалізовані комп'ютерні системи

на тему: Система ідентифікації та класифікації зображень для соціальних мереж

Виконав (-ла): студент (-ка) ІІ курсу, групи КВ-73мп

Волонтир Олександр Олегович

Науковий керівник доцент кафедри, ктн, Орлова М.М.

Рецензент д.т.н., професор, професор кафедри ОТ Кулаков Ю.О.

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)
«_ __»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Волонтиру Олександру Олеговичу

1. Тема дисертації Система ідентифікації та класифікації зображень для соціальних мереж, науковий керівник дисертації Орлова Марія Миколаївна доцент кафедри СПіСКС, к.т.н., доцент

затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації 07 грудня 2018 р.

3. Об'єкт дослідження система пошуку користувачів соціальних мереж

4. Предмет дослідження алгоритм ідентифікації та класифікації зображень

5. Перелік завдань, які потрібно розробити
   - опис предметної області досліджень та обґрунтування використання згорткової нейронної мережі;
   - методи ідентифікації та класифікації зображень;
   - модифікований алгоритм багатокластерної класифікації зображень.

6. Перелік ілюстративного матеріалу
   - Основні правила алгоритму.

– Схема нейронної мережі.
– Таблиця результатів

7. Перелік публікацій
– «Модифікований алгоритм багатокластерної класифікації зображень», IX наукова конференція магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК-2018;
– «Поєднання Machine Learning та VR», XX міжнародна науково-технічна конференція SAIT 2018;

8. Дата видачі завдання 5 вересня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення літератури за тематикою проекту	05.09.2017	
2	Аналіз існуючих рішень	20.01.2018	
3	Підготовка матеріалів першого розділу магістерської дисертації	09.03.2018	
4	Підготовка матеріалів другого розділу магістерської дисертації	30.04.2018	
5	Підготовка матеріалів третього розділу магістерської дисертації	10.09.2018	
6	Підготовка графічної частини дипломного проекту	16.10.2018	
7	Оформлення документації дипломного проекту	01.11.2018	
8	Попередній розгляд магістерської дисертації на кафедрі	26.11.2018	

Студент _____ Волонтир О.О.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Орлова М.М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми. Все частіше люди використовують соціальні мережі для полегшення процесу керування бізнесом за допомогою пошуку партнерів або людей, які мають спільні з ними інтереси.

Сьогодні існують безліч сторінок, створених в різних соціальних мережах для, наприклад, продажу певного товару, які часто шукають потенційних клієнтів за допомогою реклами або розсилання спам-повідомлень випадковим людям. Такий спосіб пошуку клієнтів або партнерів є дуже неефективним, до того ж величезна кількість людей, яким це може бути нецікавим, отримує час від часу повідомлення такого характеру.

Тому актуальною залишається задача пошуку людей за інтересами в соціальних мережах, яку можна вирішувати різними способами, одним з яких є аналіз фотографій, викладених користувачами в мережу.

Мета і задачі дослідження. Метою даної роботи є розробка системи ідентифікації та класифікації зображень для соціальних мереж, яка б дозволила зробити пошук користувачів більш ефективним. Для досягнення поставленої мети в роботі вирішуються наступні задачі.

1. Аналіз і використання сучасних методів класифікації зображень.
2. Розробка способу та системи ідентифікації та класифікації зображень, яка б дозволила шукати користувачів соціальної мережі на основі аналізу їх фотографій.
3. Аналіз та оцінка роботи розробленої системи.

Об'єктом дослідження є система пошуку користувачів соціальних мереж.

Предметом дослідження є система ідентифікації та класифікації зображень, викладених користувачами в соціальних мережах.

Методи дослідження. В роботі використовуються методи штучної нейронної мережі.

Наукова новизна одержаних результатів полягає в наступному.

1. Проаналізовано методи ідентифікації та класифікації зображень з використанням конволюційних нейронних мереж та показано, що можна ефективно використовувати існуючи моделі для вирішення поставленої задачі.
2. Запропоновано спосіб багатокластерної класифікації зображень на основі перенавчання та модифікації існуючої згорткової нейронної мережі, який дозволяє отримати високу точність розпізнавання, потребуючи менших обчислювальних ресурсів та часу на реалізацію порівняно зі створенням нового алгоритму.

Практична цінність одержаних результатів полягає в тому, що запропонована система дозволяє ефективно, з точки зору часу, шукати користувачів соціальної мережі на основі аналізу та класифікації зображень за декількома критеріями.

Апробація роботи. Основні положення і результати роботи були представлені на:

1. IX науковій конференції магістрантів та аспірантів “Прикладна математика та комп’ютинг” ПМК-2018 (Київ, 14-16 листопада 2018 р.).
2. 20-th International Conference “System Analysis and Information Technology” SAIT-2018 (Київ, 21-24 травня 2018 р.).

Публікації. За тематикою проведених досліджень опубліковано 2 наукові праці, а саме тези доповідей на конференціях.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, трьох розділів, висновків та додатків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи.

У першому розділі розглянуто теоретичні відомості з заданої теми, а також проведено аналіз методів класифікації зображень.

У другому розділі проведено аналіз методів та засобів для класифікації зображень з використанням згорткових нейронних мереж та описано модифікований метод багатокластерної класифікації.

У третьому розділі описано роботу розробленої системи та проаналізовано отримані результати класифікації зображень, викладених користувачами в соціальні мережі.

У висновках представлені результати проведеної роботи.

Ключові слова: соціальна мережа, конволюційна нейронна мережа, класифікація зображень, багатокластерна класифікація, метод перенавчання нейронної мережі.

ЗМІСТ

ВСТУП

В час розвитку інформаційних технологій суспільство звикло до того, що автоматизація будь-яких процесів швидко і міцно впроваджується в наше життя. Зараз існує багато різноманітних систем, які допомагають людям займатись своєю улюбленою справою, керувати виробничим процесом, а також відіграють значну роль у повсякденному житті. Можна з впевненістю сказати, що Інтернет є тією величезною мережею, яка надає змогу на існування таким системам. Не так давно люди створили і почали користуватись різними соціальними мережами. Спочатку їхньою метою було спілкування та підтримання звичайних стосунків в режимі онлайн. Згодом ці мережі почали використовуватись як для отримання якоїсь корисливої інформації, так і для розваг. Зараз вони також надають можливість людям полегшити процес керування бізнесом за допомогою пошуку партнерів або людей, які мають спільні з ними інтереси.

Однією з проблем, які залишаються при намаганні використовувати соціальну мережу для своїх потреб або ведення власної справи, є пошук людей, які були б зацікавленні в тому, шо пропонується. Сьогодні існують безліч сторінок, створених в різних соціальних мережах для, наприклад, продажу певного товару, які часто шукають потенційних клієнтів за допомогою реклами або розсилання спам-повідомлень випадковим людям. Такий спосіб пошуку клієнтів або партнерів є дуже неефективним, до того ж може дратувати величезну кількість людей, які отримують з періодичністю повідомлення такого характеру.

Для того щоб частково вирішити вище описану проблему пошуку людей в соціальних мережах, була поставлена задача розробити систему, яка б допомагала знаходити їх на основі аналізу та класифікації зображень, викладених в мережу користувачами. Така система дозволила б класифікувати користувачів за сферами інтересів, тим самим відкидаючи тих, хто не зацікавлений в тій чи іншій пропозиції. Це надасть змогу скоротити

час пошуку людей та зменшити обсяг непотрібних користувачам повідомлень.

РОЗДІЛ 1. ОГЛЯД МЕТОДІВ АНАЛІЗУ ТА КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

Задача класифікації зображень дуже часто є важливою частиною обробки будь-яких даних. Віднести зображення до того чи іншого класу можна за допомогою розпізнавання або виділення певного об’єкту на ньому, але іноді може знадобитись досить складний аналіз, реалізації якого можуть бути різними в залежності від самої задачі та вимог до продуктивності алгоритму, а також часу, затраченого на його вивчення та розробку. Далі будуть розглянуті різноманітні підходи для вирішення цієї задачі, які використовують методи машинного навчання.

1.1 Задача класифікації зображень

Машинне навчання набирає обертів в останні роки: автомобілі з автоматичним керуванням, ефективний пошук в Інтернеті, розпізнавання голосу та зображень. Все це поступово стає невід’ємною складовою нашого повсякденного життя. Машинне навчання – це клас методів штучного інтелекту, який дозволяє комп'ютеру працювати в режимі самонавчання без явного програмування [1]. Це дуже цікава і складна тема, яка може стимулювати майбутнє технологій.

Однією з задач, яку можуть вирішувати методи машинного навчання, є аналіз та класифікація зображень. Класифікація об'єктів є простим завданням для людини, але досить складним для комп’ютера. Класифікація зображень може включати попередню обробку зображення, виявлення, сегментацію та класифікацію об'єктів. Найважливішою складовою системи класифікації зображень є база даних, яка містить заздалегідь визначені шаблони, які порівнюються з об'єктом для класифікації до відповідної категорії.

Класифікація зображень може складатися з наступних етапів:

- підготовка даних;

- первинна обробка зображення: видалення шуму, технології корекції та ін.;
- виявлення важливих характеристик зображення;
- класифікація: зображення класифікуються в заздалегідь визначені категорії на основі виявлених важливих характеристик.

1.1.1 Порівняння методів класифікації зображень

Взагалі існує декілька передових методів класифікації цифрових зображень, серед яких є штучні нейронні мережі, метод опорних векторів, метод нечіткої міри, а також генетичні алгоритми, основні переваги та недоліки яких наведені в таблиці 1.1 [2].

Таблиця 1.1 – Переваги та недоліки методів класифікації зображень

Алгоритм машинного навчання	Переваги	Недоліки
Нейронна мережа	• Може бути використаний для класифікації або регресії. • Стійкий до шуму у вхідних даних. • Підтримка багатокластерної класифікації	• Оптимальна структура мережі може бути визначена лише експериментальним шляхом
Метод опорних векторів	• Низька ймовірність перенавчання. • Легкий контроль складності правила прийняття рішення та частоти помилок	• Важко визначити оптимальні параметри даних для тренування. • Складна структура алгоритму

Метод нечіткої міри	• Для опису властивостей можна ідентифікувати різні стохастичні відносини	• Попередні знання дуже важливі для отримання результатів високої точності
Генетичний алгоритм	• Може використовуватися в класифікації та виборі ознак	• Складний алгоритм розробки функції оцінювання. • Ускладнення, пов'язані з представленням тренувальних даних

Співвідношення переваг та недоліків у нейронних мереж, як показано в таблиці 1.1, найбільше порівняно з іншими методами класифікації зображень. Більш того, обмеження цього алгоритму машинного навчання зустрічається в багатьох інших алгоритмах. Також майже всі алгоритми вимагають досить складної підготовки даних для тренування або навчання. Оскільки нейронні мережі здатні до багатокластерної класифікації, що знадобиться при обробці фотографій користувачів соціальної мережі, а також мають багато засобів для обробки та запобігання шуму в зображеннях, вони є оптимальним рішенням для поставленої задачі.

1.2 Нейронні мережі

Нейронна мережа – це алгоритм машинного навчання, який будується на принципі організації та функціонування біологічних нейронних мереж [3].

Нейронні мережі використовують інший підхід до вирішення проблем, ніж звичайні комп'ютери. У звичайних комп'ютерах використовується

алгоритмічний підхід, тобто виконується набір інструкцій для вирішення проблеми. Відомо, що комп'ютери не можуть вирішити проблему, якщо не будуть відмічені конкретні кроки, які він повинен виконати. Це обмежує проблему вирішення можливостей звичайних комп'ютерів до проблем, які ми вже розуміємо і вміємо вирішити. Але комп'ютери були б набагато кориснішими, якщо б вони могли виконувати те, що людина не знає точно як робити.

Нейронні мережі обробляють інформацію подібно тому, як це робить людський мозок. Мережа складається з великої кількості взаємопов'язаних елементів обробки (нейронів), що працюють паралельно для вирішення конкретної проблеми. Нейронні мережі навчаються на прикладі. Вони не можуть бути запрограмовані для виконання конкретного завдання. Приклади мають бути підібрані таким чином, щоб мережа навчалась з мінімальною кількістю помилок, а це означає, що вони мають бути унікальними й охоплювати всі можливі ситуації, які можуть зустрітись під час використання готової моделі. Недолік полягає в тому, що мережа виявляє, як вирішити проблему сама по собі, її операція може бути непередбачуваною.

З іншого боку, звичайні комп'ютери використовують когнітивний підхід до вирішення проблем: проблема повинна бути відома і зазначена в невеликих однозначних інструкціях. Ці інструкції потім перетворюються на програму мови високого рівня, а потім на машинний код, який комп'ютер може зрозуміти, що робить процес обробки цілком передбачуваним. Якщо щось трапиться не так, це пов'язано з несправністю програмного чи апаратного забезпечення.

Нейронні мережі та звичайні алгоритмічні комп'ютери не конкурують, але доповнюють один одного. Існують такі задачі, для вирішення яких застосовується алгоритмічний підхід, наприклад, арифметичні операції, та задачі, які більше підходять для нейронних мереж. Більш того, велика кількість завдань вимагає систем, які використовують комбінацію двох

підходів для максимально ефективної роботи.

Концепція нейронної мережі виникла при спробі імітувати процеси, що відбуваються в мозку людини, ще у 1943 році. Вона складається з окремих одиниць, що називаються нейронами, які розташовані в серіях груп або шарів (рис. 1.1). Нейрони в кожному шарі з'єднуються з нейронами наступного шару. Дані проходять від вхідного шару до вихідного уздовж таких сполук. Кожен окремий вузол виконує простий математичний розрахунок, а далі передає отриманий результат всім вузлам, з якими він зв’язаний. Цей результат перед потраплянням до наступного нейрону множиться на вагу з’єднання. Ваги на самому початку тренування мережі задаються випадковим чином, після чого змінюються в процесі навчання таким чином, щоб зробити результуюче значення моделі мінімально відмінним від очікуваного.

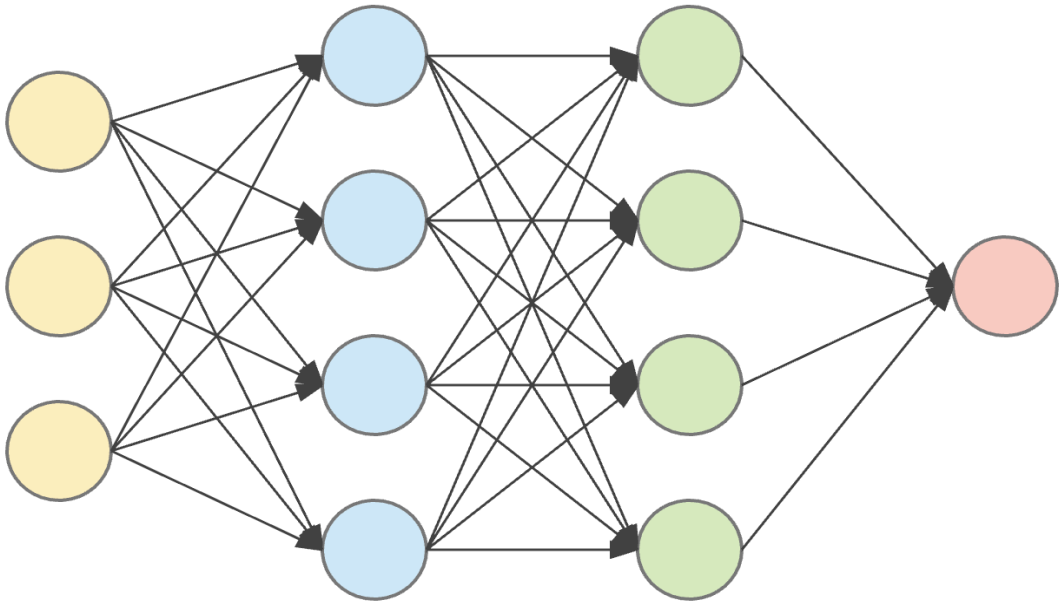

Рисунок 1.1 – Схема нейронної мережі

Існує декілька типів штучних нейронних мереж, серед яких є одношарові або багатошарові мережі прямого поширення та рекурентні. Для вирішення задачі класифікації зображень використовується перший тип, а другий – найчастіше для вирішення задач розпізнавання рукописного тексту

або розпізнавання голосу. Тому розглянемо архітектуру та принципи роботи нейронної мережі прямого поширення [3].

1.2.1 Нейронна мережа прямого поширення

Нейронні мережі прямого поширення (рис. 1.2) дозволяють сигналам проходити тільки в одному напрямку – від вхідного шару до вихідного. Немає зворотного зв'язку (циклів), тобто вихід будь-якого шару не впливає на той самий шар. Нейронні мережі прямого поширення в основному тренуються за методикою контрольованого навчання, коли дані, які будуть поступати на вхід, не є ні послідовними, ні залежними від часу.

Контрольоване навчання – це підхід, що використовується в машинному навчанні, який визначає функцію за допомогою аналізу даних, помічених будь-якими мітками [3]. Наприклад, в алгоритмі машинного навчання, який визначає, чи повідомлення є спамом чи ні, набір даних для тренувань повинен включати повідомлення, позначені як “спам”, а також повідомлення, позначені як “не спам”, для того, щоб навчитись визначати різницю.

Нейронні мережі прямого поширення широко використовуються при розпізнаванні образів. Цей тип організації також називається “знизу вгору” або “зверху вниз”. Метою мережі прямого поширення є апроксимація деякої функції f. Наприклад, $y = f(x)$ визначає відповідність між вхідним значенням x та якоюсь категорією y. Мережа визначає $y = f(x;\theta)$ і вивчає значення параметрів θ, що призводить до найкращого наближення функції. Саме таку задачу необхідно вирішити для класифікації зображень.

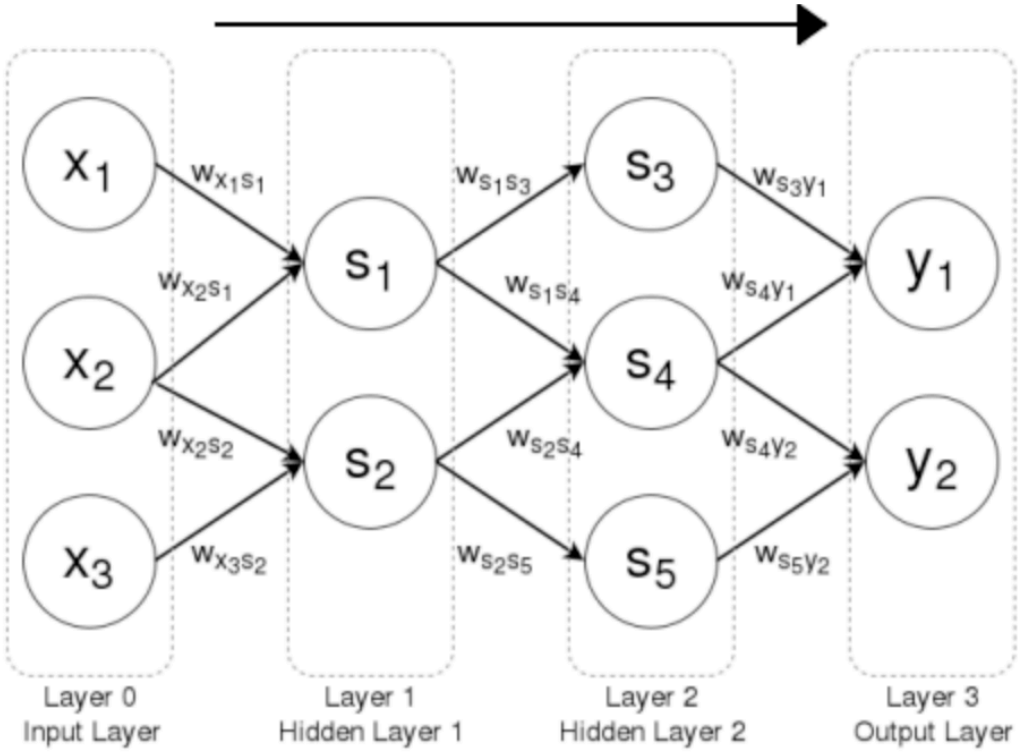

Рисунок 1.2 – Схема нейронної мережі прямого поширення

Найпоширеніший тип штучної нейронної мережі складається з трьох груп або шарів: “вхідний” шар, підключений до “прихованого” шару (або декількох), за яким слідує “вихідний” (рис. 1.2).

1.3 Принципи роботи нейронної мережі прямого поширення

Штучний нейрон (рис. 1.3) – це пристрій з багатьма входами та одним виходом [3]. Нейрон має два режими роботи: режим тренувань і режим використання. У тренувальному режимі нейрон може бути навчений як слід йому реагувати на певні шаблони вводу. У режимі використання введений шаблон розпізнається, пов'язане з ним вихідне число стає поточним виходом. Якщо шаблон входу не належить до списку шаблонів, які використовувались під час тренування, правило навчання використовується для визначення того, що буде на виході.

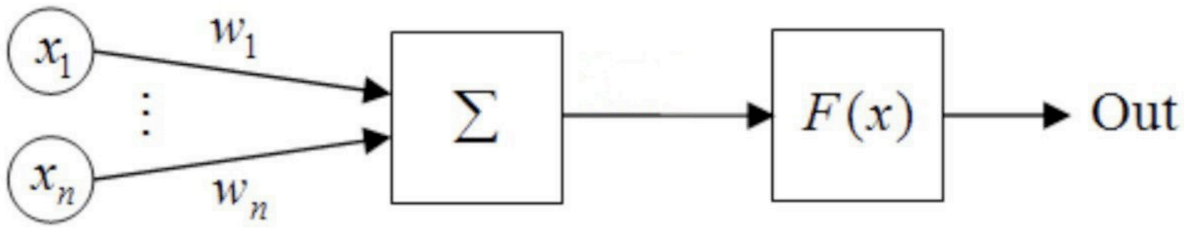

Рисунок 1.3 – Штучний нейрон

На рисунку 1.3 різні входи до мережі, представлені математичним символом x_i. Кожен з цих входів множиться на вагу з'єднання w_i. У найпростішому випадку ці значення підсумовуються, подаються на функцію передачі (функцію активації) для генерації результату, і цей результат надсилається на вихід. Існує можливість, що різні мережеві структури використовують різні сумарні функції, а також різні функції передачі.

Штучний нейрон складається з семи основних компонентів. Ці компоненти дійсні та використовуються в залежності від того, в якому шарі розташований нейрон: в шарі введення, виведення або в “прихованих”.

Перша складова нейрона – це фактори ваг. Нейрон у звичайному випадку має багато входів. Кожен вхід має свою відносну вагу, добуток якої на вхідне значення передається до функції підсумовування. Деякі входи стають важливішими, ніж інші, щоб мати більший вплив на оброблювальний елемент, оскільки їх значення об'єднуються для формулювання єдиної відповіді. Вага – адаптивний коефіцієнт, який визначає інтенсивність вхідного сигналу, зареєстрованого штучним нейроном. Вони є мірою сили підключення вхідного сигналу і можуть бути змінені у відповідь на різні тренувальні набори та відповідно до специфічної топології мережі або її правил навчання.

Після множення вхідного значення на вагу воно передається до функції підсумовування. Входи та відповідні їм ваги – це вектори, які можуть бути

представлені як $(x_1, x_2, ..., x_n)$ та $(w_1, w_2, ..., w_n)$. Загальний вхідний сигнал – це скалярний добуток цих двох векторів. Слід зауважити, що функція підсумовування може бути більш складною, ніж просто вагова сума вхідних значень. Коефіцієнти введення та ваг можуть бути об'єднані різними способами перед потраплянням до передавальної функції. Специфічний алгоритм поєднання нейронних входів визначається обраною мережевою архітектурою та парадигмою.

Найважливішою складовою штучного нейрона є передавальна функція. Результат функції підсумовування перетворюється на вихідне значення за допомогою алгоритму, реалізованого на основі передавальної функції. У функції передачі підсумовування можна порівняти з певним порогом, щоб визначити вихідне число. Якщо сума перевищує порогове значення, оброблювальний елемент (нейрон) генерує сигнал передачі результату до наступного шару, інакше сигнал не генерується. Обидва типи відповідей є значними. Порогова або передавальна функція, як правило, нелінійна. Лінійні функції обмежені, оскільки результат пропорційний вхідним параметрам.

Результат функції передачі може проходити через додаткові процеси, які мають назви масштабування і обмеження. Масштабування означає множення значення попередньої функції на відповідний коефіцієнт, а також додавання зміщення. Обмеження – це механізм, який гарантує, що масштабований результат не перевищує верхню або нижню межу. Це обмеження є додатковим до тих, які, можливо, містяться в первісній передавальній функції.

Кожний оброблювальний елемент генерує один вихідний сигнал, який може бути переданий до сотні інших нейронів. Зазвичай вихід безпосередньо еквівалентний результату функції передачі. Деякі мережеві топології змінюють результат функції передачі для забезпечення змагання (або конкуренції) між сусідніми елементами обробки. Нейрони можуть змагатись

між собою. По-перше, змагання визначає, який штучний нейрон буде активним або генерує вихідний сигнал. По-друге, це допомагає визначити, який елемент обробки буде брати участь у процесі навчання або адаптації.

Останніми двома складовими нейрона є функція помилки та зворотного розповсюдження, а також функція навчання, яка являє собою будь-яке правило. У більшості навчальних мереж різниця поточного виходу та бажаного виходу розраховується як помилка, яка потім перетворюється функцією помилки відповідно до певної архітектури мережі. Більшість базових архітектур використовують цю помилку безпосередньо, але деякі підносять її до квадрату, зберігаючи свій знак, деякі – до кубу, інші парадигми змінюють помилку відповідно до їх конкретних цілей. Помилка поширюється назад до попереднього шару. Зазвичай це значення, після масштабування за допомогою функції навчання, множиться на кожну з вхідних ваг, щоб змінити їх до наступного циклу навчання. В мережах прямого поширення навчання відбувається в основному за алгоритмом зворотного розповсюдження [4].

1.3.1 Алгоритм зворотного розповсюдження

Алгоритм зворотного розповсюдження дуже важливий для швидкого навчання великих нейронних мереж (з великою кількістю шарів). Тому необхідно розуміти принципи його роботи, які схематично і досить примітивно зображені на рисунку 1.4.

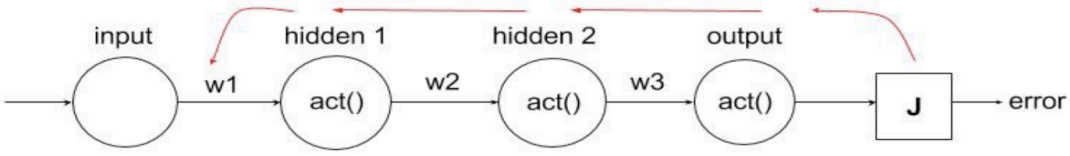

Рисунок 1.4 – Схема роботи алгоритму зворотного розповсюдження

Для того, щоб розібратись зі схемою, представленою вище, уявимо просту мережу прямого поширення, яка складається з одного вхідного, одного вихідного та двох прихованих шарів з двома нейронами в кожному. Вузли в сусідніх прихованих шарах мають з'єднання кожний з кожним, а ваги цих з'єднань є параметрами мережі і, нехай, позначаються як w_{i_j}. Кожний вузол має вхідне значення x_i, функцію активації *f(x)* і вихідне значення y_i, яке є результатом функції активації. *f(x)* має бути нелінійною функцією, інакше нейронна мережа зможе вивчати тільки лінійні властивості. Часто використовуваною такою функцією є сигмоїдна, але детальніше про функції активації буде розказано в наступному розділі.

Алгоритм зворотного розповсюдження використовується для того, щоб автоматично визначити всі ваги в мережі, маючи дані для навчання, так що результат роботи моделі (прогнозування) y_{out} завжди буде найближчим до очікуваного y_{tar} для будь-яких вхідних значень. Щоб виміряти, наскільки прогнозоване значення відрізняється від очікуваного, використовується функція помилки *J*, яка найчастіше визначається формулою:

$$J = \frac{1}{2}(y_{out} - y_{tar})^2 \ .$$

Для того, щоб обчислити результуюче значення нейронної мережі застосовуються дві основні формули:

$$x_j = \sum_i^j w_{i_j} y_i + b_j \ ,$$

яка вираховує вхідне значення кожного нейрона як вагу помножену на вихід попереднього нейрона з додаванням зміщення, яке задається перед початком навчання, та $y = f(x)$, що є функцією активації в окремому вузлі, результатом якої і буде вихідне значення окремого нейрона.

Алгоритм зворотного розповсюдження вирішує, як треба змінити кожну вагу мережі після порівняння прогнозованого результату з очікуваним

для конкретного прикладу. Для цього потрібно обчислити $\frac{dJ}{dw_{i_j}}$, що покаже, як змінюється помилка у результаті зміни кожного ваги. Як тільки отримано ці похідні, можна приступати до оновлення ваг за допомогою простого правила:

$$w_{i_j} = w_{i_j} \text{-} \alpha \frac{dJ}{dw_{i_j}} ,$$

де α являє собою додатну константу, яка потребує емпіричного підбору.

Правило оновлення дуже просте: якщо помилка зменшується при збільшенні ваги ($\frac{dJ}{dw_{i_j}} < 0$), то необхідно збільшити вагу, в іншому випадку, якщо помилка зростає, коли вага збільшується ($\frac{dJ}{dw_{i_j}} > 0$), необхідно зменшити вагу.

Для спрощення обчислень $\frac{dJ}{dw_{i_j}}$, потрібно додатково зберігати для кожного вузла значення ще двох похідних: $\frac{dJ}{dx}$ та $\frac{dJ}{dy}$, які показують, як помилка змінюється із загальним вхідним та вихідним значенням відповідно.

Як тільки обчислені всі похідні, починається їх зворотне поширення. З урахуванням функції помилки маємо наступне:

$$\frac{dJ}{dy_{out}} = y_{out} \text{-} y_{tar} \ .$$

А маючи $\frac{dJ}{dy}$, можна отримати $\frac{dJ}{dx}$, використовуючи правило ланцюга:

$$\frac{dJ}{dx} = \frac{dy}{dx}\frac{dJ}{dy} = \frac{d}{dx} f(x) \frac{dJ}{dy} \ .$$

Після взяття похідної функції помилки по x, можна обчислити похідну по w_{i_j}:

$$\frac{dJ}{dw_{i_j}} = \frac{dx_j}{dw_{i_j}}\frac{dJ}{dx_j} = y_i \frac{dJ}{dx_j} \ .$$

І, використовуючи правило ланцюга, можна також отримати $\frac{dJ}{dy}$ з попереднього шару:

$$\frac{dJ}{dy_i} = \sum_{j \in i} \frac{dx_j}{dy_i} \frac{dJ}{dx_j} = \sum_{j \in i} w_{i_j} \frac{dJ}{dx_j} \ .$$

Все, що залишається зробити, це повторити обчислення, використовуючи попередні три формули, поки не будуть отримані всі похідні функції помилки.

1.3.2 Правило навчання нейронної мережі

Правило (або алгоритм) навчання є важливою концепцією в нейронних мережах і пояснюється їх високою гнучкістю. Цей алгоритм визначає, що буде на виході нейрона в залежності від вхідних параметрів. Це стосується будь-яких параметрів, а не тільки тих, які поступали на вхід під час навчання (тренування). Простим прикладом роботи такого правила є наступні дії. Якщо вхідною комбінацією є та, що відрізняється від всіх попередньо розглянутих нейроном, то буде визначатись вихідне число (наприклад, 1 або 0) шляхом порівняння окремих вхідних значень з найближчими комбінаціями. Наприклад, нейрон з трьома входами має на виході 1, коли вхід (x1, x2 і x3) становить 111 або 101, і 0, коли вхід дорівнює 000 або 001. Тому, перед застосуванням алгоритму, таблиця істинності виглядає наступним чином (табл. 1.2).

Таблиця 1.2 – Таблиця істинності нейрона з трьома входами

x1	x2	x3	Out
0	0	0	0
0	0	1	0
0	1	0	0/1
0	1	1	0/1
1	0	0	0/1
1	0	1	1

x1	x2	x3	Out
1	1	0	0/1
1	1	1	1

Для розгляду роботи алгоритму навчання можна взяти шаблон 010. Він відрізняється від 000 одним вхідним значенням, від 001 – двома, віл 101 – трьома і від 111 – також двома. Тому найближчим шаблоном, який мінімально відрізняється від обраного, є 000, що має 0 на виході. Таким чином, правило вимагає, щоб нейрон мав 0 на виході, коли на вході 001. З іншого боку, 011 однаково віддалений від двох шаблонів, які мають різні виходи, і, таким чином на виході залишається невизначений стан (0/1).

Застосовуючи такий підхід, можна перетворити таблицю істинності на наступну (табл. 1.3).

Таблиця 1.3 – Таблиця істинності нейрона з трьома входами після застосування правила навчання

x1	x2	x3	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0/1
1	0	0	0/1
1	0	1	1
1	1	0	1
1	1	1	1

Важливим застосуванням нейронних мереж є розпізнавання образів. Розпізнавання образів може бути реалізовано за допомогою нейронної мережі, яка була підготовлена належним чином. Під час тренувань мережа навчається поєднувати результати з шаблонами введення. Коли мережа

використовується, вона визначає вхідний шаблон і намагається вивести пов'язаний з ним вихідний. Коли вхідний шаблон не має пов’язаного з ним вихідного, мережа аналізує його і видає результат, який відповідає навчальному вхідному шаблону, що менше відрізняється від заданого.

1.4 Конволюційні нейронні мережі та класифікація зображень

Конволюційні нейронні мережі – це особлива архітектура штучних нейронних мереж, запропонована Яном Лекуном у 1988 році [5]. Одним з найпопулярніших застосувань цієї архітектури є класифікація зображень. Наприклад, Facebook використовує їх для автоматичного проставлення тегів, Amazon – для створення рекомендацій щодо продукту, а Google – для пошуку фотографій користувача.

Задачею класифікації зображень є визначення категорії, до якої вони належать. Людина може з легкістю впоратись з такою задачею на відміну від комп’ютера, оскільки кожне зображення для нього представляється як матриця значень пікселів. Тому необхідно відповідним чином оброблювати ці значення. Саме це і робить конволюційна нейронна мережа, яка складається з наступних шарів:

- вхідний шар, який приймає зображення, якщо необхідно, змінює їх розмір для передачі до наступних шарів, де відбувається виявлення об'єктів;
- конволюційні (або згорткові) шари, які відіграють роль фільтрів для зображень, виділяючи певні об’єкти на них, а також використовуються для розрахунку міри відповідності під час тестування;
- агрегувальні шари, які об'єднують виходи кластерів нейронів одного шару до одного нейрону наступного шару. Служать для поступового зменшення просторового розміру представлення, для зменшення

кількості параметрів та обсягу обчислень в мережі, а отже, і для контролю за перенавчанням;

- шар зрізаних лінійних вузлів, який кожне від'ємне число, отримане з попередніх шарів, замінює на 0, що допомагає зберігати стабільність в нейронній мережі, запобігаючи варіювання значень від 0 до нескінченності;
- повноз'єднаний шар, який приймає зображення з високим рівнем фільтрації та виконує функцію класифікації – віднесення його до певної категорії.

Головним шаром згорткової нейронної мережі є конволюційний (згортковий), тому цей вид мережі і має таку назву. Шар складається з набору фільтрів, за допомогою яких відбувається прохід по вхідним даним і мережа навчається. Протягом прямого проходу кожен фільтр здійснює згортку за шириною та висотою вхідної ємності, обчислюючи скалярний добуток даних фільтру та вхідних, які представленні значеннями кожного з пікселів зображення, і формуючи двовимірну карту збудження цього фільтру (рис. 1.5).

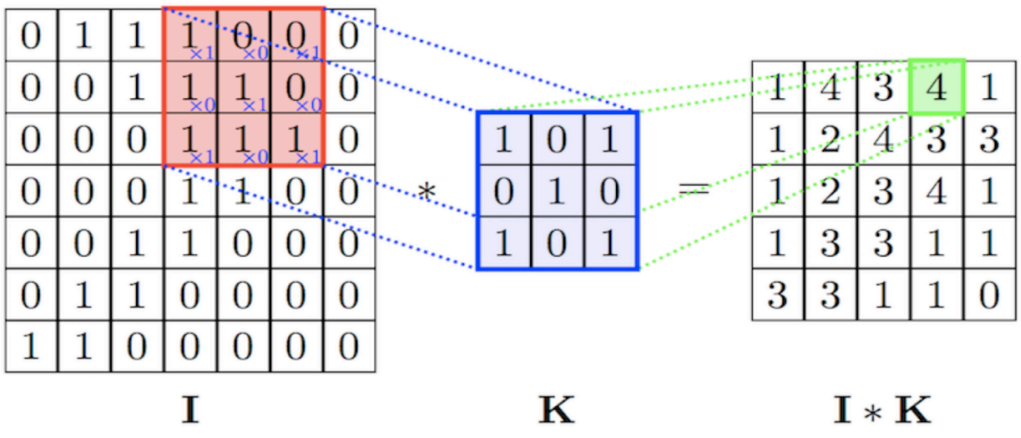

Рисунок 1.5 – Схема роботи фільтру згорткового шару

1.4.1 Метод перенавчання конволюційної нейронної мережі

В задачах класифікації зображень зазвичай можна побачити, що категорії цих зображень досить специфічні та унікальні і, можливо, не були враховані при розробці та навчанні відомих нейронних мереж. Тому рішенням, яке змогло б забезпечити досить велику точність результату на виході, є розробка власної моделі з нуля та навчання її на даних, які підходять до цієї окремої задачі. Але створення спеціальної моделі може призвести до використання великих обчислювальних ресурсів, чималого об’єму даних для навчання, а також саме навчання може зайняти немало часу. До того ж вже існують готові універсальні моделі, які навчені класифікувати зображення різних категорій. Однією з таких моделей є Inception V3, при розробці якої була використана база даних ImageNet з численною кількістю зображень (рис. 1.6).

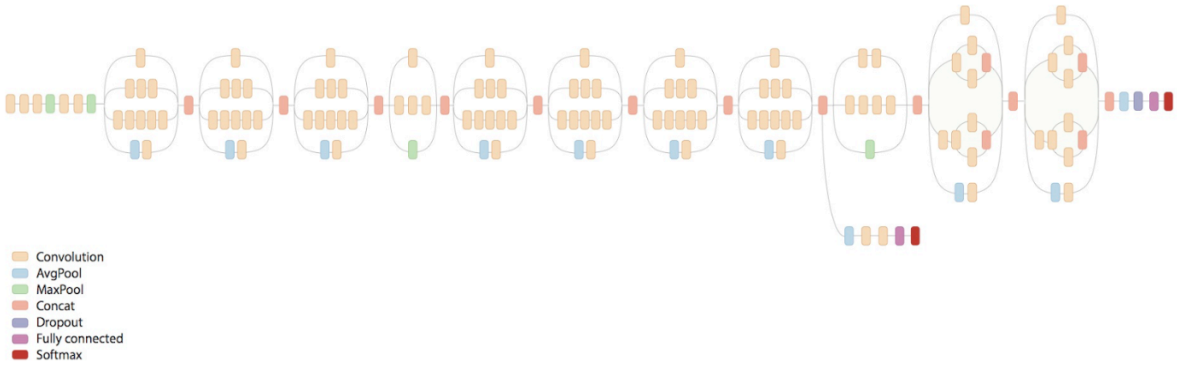

Рисунок 1.6 – Схема моделі Inception V3

Загальним та ефективним підходом до класифікації зображень специфічних категорій є використання попередньо підготовленої мережі (або моделі). Попередньо підготовлена мережа – це збережена модель, яка була навчена класифікувати образи на великому наборі даних, як правило, при вирішенні задачі класифікації зображень, категорії яких лежать в дуже

великому діапазоні. Якщо цей оригінальний набір даних достатньо великий і достатньо загальний, то ієрархія ознак, вивчених попередньо підготовленою мережею, може виявитися корисною для вирішення багатьох різних проблем, пов’язаних з комп’ютерним зором, хоча ці нові задачі можуть включати зовсім інші класи, ніж ті, що були враховані при розробці якоїсь конкретної моделі.

Процес реалізації методу перенавчання включає в себе наступні кроки [6].

1. Вибір попередньо навченої моделі. Взагалі існує досить велика кількість таких моделей, наприклад, AlexNet, ResNet, ResNeXt, Inception (GoogleNet) та інші.
2. Розділення старої архітектури моделі на дві частини:
    - всі приховані шари нейронної мережі з їх структурою (з'єднаннями) і раніше вивченими вагами, що будуть скопійовані в нову модель;
    - останній шар нейронної мережі, пов'язаний зі старою моделлю, який виконує фактичну класифікацію в одному з класів і який буде знехтуваний в новій моделі.
3. Підготовка нового набору навчальних даних (зображення, що відносяться до відповідного класу, потрібні для нової моделі).
4. Обчислення вихідних значень для нового набору тренувальних зображень після проходження через першу частину нейронної мережі (ту, яка взята з попередньо навченої). Розраховане чисельне значення в останньому шарі оригінальної моделі буде називатися “вузьким місцем”.
5. Додавання нового повноз’єднаного шару, який тепер стане останнім шаром нової моделі нейронної мережі. Цей новий шар буде розрахувати ймовірність приналежності даного зображення до даного класу.

6. Навчання нового (повноз'єднаного) шару з попередньо розрахованими “вузькими місцями”, що є вхідними значеннями, а також набором нових “істинних” міток, які позначають справжні класи навчальних зображень.

Таким чином, проведений аналіз показав, що на сьогоднішній день немає універсальних класифікаторів зображень і що кожна окрема задача повинна вирішуватись за допомогою створення нового алгоритму або модифікації існуючого.

Висновки до розділу 1

1. Проаналізовано існуючі підходи до класифікації зображень і показано, що існуючі рішення не є універсальним розв’язком задачі класифікації.
2. Запропоновано використовувати підхід для вирішення поставленої задачі, який полягає в використанні існуючої згорткової нейронної мережі та методу перенавчання на її основі, тобто заміни лише одного з багатьох її шарів, що потребує менших обчислювальних ресурсів та часу на реалізацію порівняно зі створенням нового алгоритму.

РОЗДІЛ 2. АНАЛІЗ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ТА ТЕХНОЛОГІЙ

В попередньому розділі визначено, що для вирішення задачі класифікації зображень використовуються конволюційні (згорткові) нейронні мережі. Також розглянуто підхід, в якому використовуються відомі попередньо навчені моделі, для яких застосовується метод перенавчання. В цьому розділі будуть розглянуті та порівняні різні моделі, після чого буде обрана одна з них на основі показників ефективності використання (час розробки, потрібність у додатковій обробці зображень) та точності видачі бажаного результату.

Майже всі архітектури конволюційної нейронної мережі дотримуються тих самих загальних принципів проектування, що говорять про застосовунння згорткових шарів до вхідного об’єкту, зменшення просторових розмірів з одночасним збільшенням кількості відповідних ознак.

В той час, як класичні мережеві архітектури складаються просто з набору згорткових шарів, сучасні архітектури вивчають нові та інноваційні способи побудови згорткових шарів таким чином, щоб дозволити більш ефективне навчання мережі. Майже всі ці архітектури базуються на одному модулі, який використовується по всій мережі. Вони являються загальними принципами проектування, які використовуються та адаптуються для вирішення різних задач комп'ютерного зору. Ці архітектури є загальними екстракторами ознак, які можуть бути використані для класифікації зображень, ідентифікації об'єктів, сегментації зображень та багатьох інших передових задач.

Класичні мережеві архітектури (моделі):

- LeNet-5
- AlexNet
- VGG 16
- ZFNet

Сучасні мережеві архітектури:

- GoogLeNet
- Inception v3
- ResNet
- ResNeXt
- DenseNet

При розробці конкретної моделі або архітектури є критичним досягнення високих показників у видачі результату (класифікації), тобто модель повинна працювати настільки точно, наскільки це можливо. Але точність залежить не тільки від архітектури розроблюваної мережі, але і від кількості даних, доступних для її навчання. А тому всі згадані вище моделі навчаються на одному й тому ж наборі даних, який називається ImageNet, що дає змогу їх порівнювати за різними ознаками [7].

ImageNet – це база даних зображень, яка все ще регулярно поповнюється, і в даний час містить 14 192 122 зображень з 21841 різною категорією. З 2010 року в ImageNet проводиться щорічний конкурс в сфері комп'ютерного зору, де учасникам надається 1,2 мільйона зображень, що належать до 1000 різних класів з набору даних ImageNet. Отже, кожна мережева архітектура використовує цю базу даних для тренування та в кінці змагання може надати показники своєї роботи.

Що стосується конструкції нейронних мереж, то тенденція розроблювати їх в глибину зростає в останні роки (рис. 2.1). Лише декілька років тому мережі могли мати приблизно дванадцять шарів, що розташовані в глибину, а зараз не дивно, що зустрічаються такі, які містять навіть сотні шарів. Для багатьох програм, найбільш відомими з яких є класифікація об'єктів, чим глибше нейронна мережа, тим краще продуктивність. Тобто, якщо використовувати більше шарів, то мережі можуть належним чином тренуватись (навчатись). Враховуючи вищесказане, запропоновано розглянути та проаналізувати роботу трьох підготовлених моделей, що мають різну архітектуру, а саме: AlexNet, ResNet та Inception v3. Кожна з них долає обмеження традиційного мережевого дизайну різними способами.

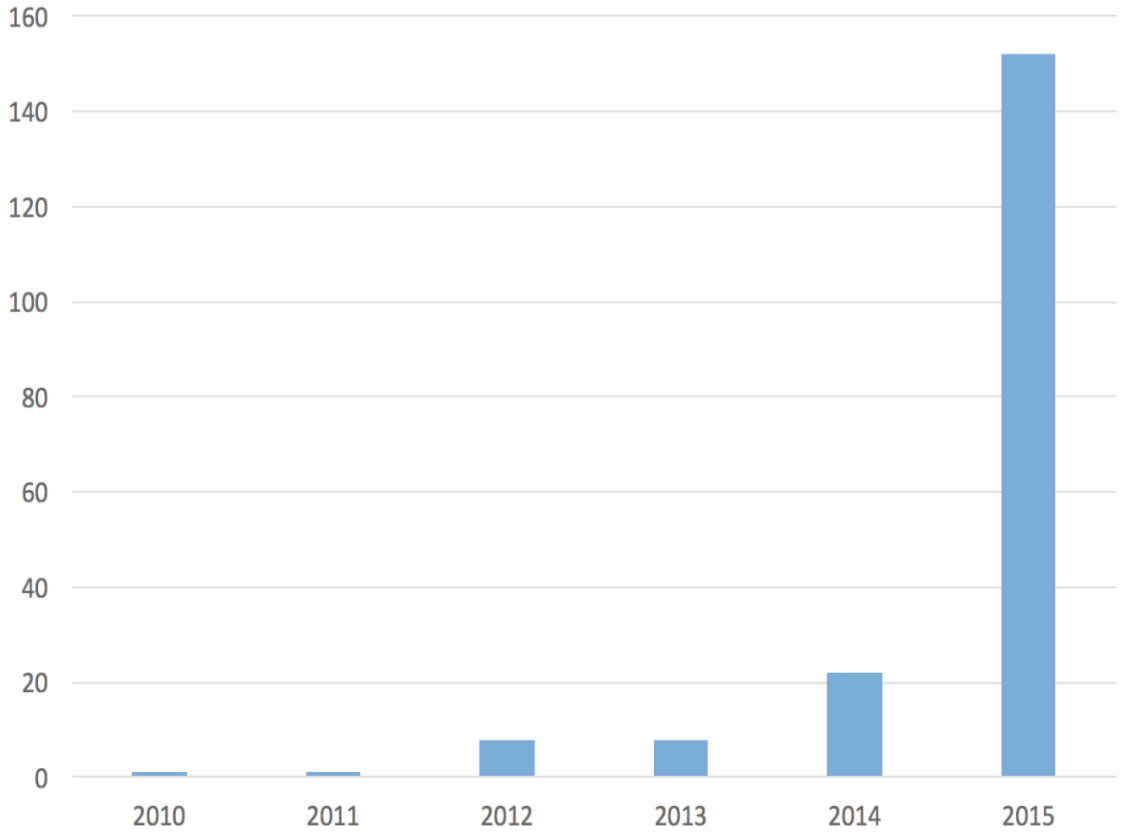

Рисунок 2.1 – Графік зростання глибини нейронних мереж (кількість шарів) переможців змагань ImageNet

Більшість конволюційних нейронних мереж мають величезні вимоги до пам'яті та обчислень, особливо під час навчання. Отже, це є досить важливою проблемою. Також, розмір кінцевої навченої моделі має значення у випадку коли, наприклад, потрібно розгорнути модель для локальної роботи на не дуже потужному пристрої. Як можна здогадатися, щоб забезпечити більшу точність, потрібна модель з більшою обчислювальною потужністю. Отже, завжди існує компроміс між точністю роботи мережі та обчисленнями, які використовуються під час її роботи.

Окрім цього, існує багато інших факторів, таких як легкість навчання, здатність мережі до узагальненості. Тому пропонується розглянути різні архітектурні рішення для подальшого їх порівняння та обрання одного з них для вирішення задачі багатокластерної класифікації зображень.

2.1 Мережева архітектура AlexNet

Ця архітектура була однією з перших конволюційних нейронних мереж, яка значно підвищила точність класифікації зображень з бази даних ImageNet порівняно з традиційними методиками [8]. Вона складається з 5 згорткових шарів, за якими слідують 3 повністю зв’язані шари, як показано на рисунку 2.1.

AlexNet використовує лінійний блок випрямлення (ReLU) для нелінійної частини замість функції тангенса (Tanh) або сигмоїду (Sigmoid), які були стандартом для традиційних нейронних мереж. Всі вищезгадані функції є функціями активації нейронної мережі або окремого її блоку та будуть розглянуті та проаналізовані далі в цьому розділі.

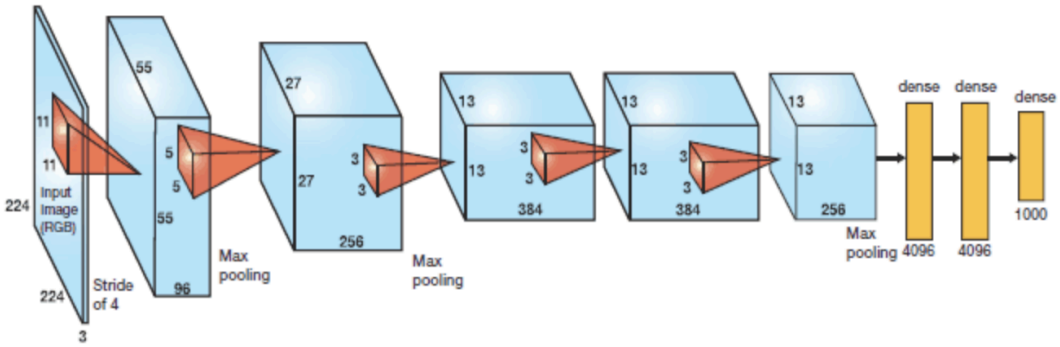

Рисунок 2.1 – Архітектура моделі AlexNet

ReLU представлена функцією $f(x) = max(0,x)$. Перевага ReLu над сигмоїдом полягає в тому, що навчання проходить набагато швидше. Проблема в тому, що похідна сигмоїда стає дуже маленькою в зоні насичення, і тому оновлення ваг майже зникає. Це називається проблемою зникання градієнту. У мережі шар випрямлення ReLu поміщається після кожного з конволюційних та повністю з'єднаних шарів, що усуває проблему призупинення оновлення ваг.

Ще однією проблемою, яку вирішила ця архітектура, було зменшення

ймовірності перенавчання, використовуючи шар виключення (dropout) після кожного повністю з'єднаного. Цей шар на виході має ймовірність *p,* яка називається коефіцієнтом виключення і застосовується для кожного з нейронів окремо. Він вимикає активацію з ймовірністю *p*, як показано на рисунку 2.2. Суть полягає в тому, щоб в процесі роботи мережі виділити підмережу, для якої буде здійснятись навчання. Для цього використовується шар виключення, який випадковим чином обирає підмережу для подальшого навчання. Під час тестування він не використовується.

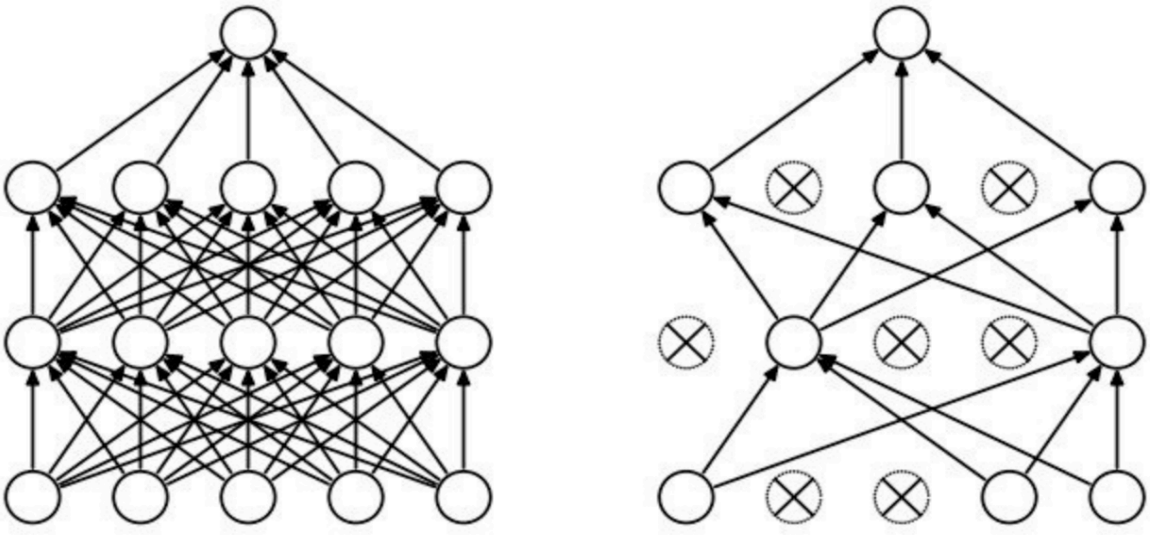

Рисунок 2.2 – Нейронна мережа до та після застосування шару виключення (dropout)

Завдяки шару виключення відбувається вимкнення активації деяких нейронів, набір яких являє собою нову, відмінну від інших, архітектуру. Всі ці утворені підмережі навчаються паралельно. Значення ваги задається для кожної з них окремо, а сума ваг дорівнює 1. Для *n* нейронів, приєднаних до шару виключення, кількість утворених підмножин становитиме 2^n.

Скажімо, результатом опрацювання зображення певного шару мережі, як правило, є вектор [0.2, 0.5, 1.3, 0.8, 1.1] для деяких вхідних даних під час тренування. Після застосування шару виключення цей вектор буде мати декілька нульових значень, розподілених випадковим чином, наприклад, [0,

0,5, 1,3, 0, 1,1]. Таким чином, це означає, що результат стає усередненим значенням отриманих від всіх утворених підмножин прогнозувань. Це забезпечує урегульованість структурованої моделі, яка допомагає уникнути перенавчання мережі. Інша користь цього шару полягає в тому, що оскільки нейрони обираються випадковим чином, вони, як правило, є незалежними один від одного, що дозволяє їм виявляти ознаки незалежно від інших нейронів.

2.2 Мережева архітектура Inception v3

Розробники моделі GoogleNet перші подумали про те, що більшість активацій у нейронній мережі є або непотрібними (нульове значення), або надмірними через кореляції між ними. Тому найефективніша архітектура глибокої мережі на їх думку має розсіяний зв'язок між активаціями, що означає, що, наприклад, всі 512 вихідних каналів не будуть мати з'єднання з усіма 512 вхідними каналами. Існують способи зменшити число таких з'єднань, що називається розсіюванням ваг або з'єднань.

Розробники GoogleNet створили модуль, що називається модулем запуску (Inception), який наближається до розрідженої конволюційної нейронної мережі з нормальною щільною конструкцією. На основі цього модулю згодом була створена модель Inception v3 [9]. Оскільки лише невелика кількість нейронів ефективно впливають на роботу моделі, як зазначалося вище, ширина або число згорткових фільтрів певного розміру ядра є також малим значенням. Також в моделі використовуються згортки різних розмірів для виявлення ознак у різних масштабах (5x5, 3x3, 1x1). Іншою вирішальною відмінністю цього модуля є наявність в ньому так званих “вузьких шарів” (згорток 1x1 розміру). Це допомагає значно зменшити вимоги до обчислень, що пояснюється далі.

Візьмемо перший модуль Inception v3 моделі для прикладу, який має

192 канали у якості вхідних даних. Мережа має всього 128 фільтрів розміру 3x3 і 32 фільтра розміром 5x5. Порядок обчислень для фільтрів 5x5 становить 25x32x192, що може бути суттєвим, оскільки із зростанням ширини мережі зростає і кількість фільтрів розміру 5x5. Для того, щоб уникнути цього, модуль Inception використовує згортки розміру 1x1 перед застосуванням ядер більшого розміру, щоб зменшити розмір вхідних каналів. Отже, в першому модулі запуску вхідні дані спочатку надсилаються в згортки 1x1 з лише 16 фільтрами, перш ніж вони будуть подаватись в згортки 5x5. Це зменшує порядок обчислень до 16x192 + 25x32x16. Всі ці зміни змушують мережу мати велику ширину і глибину.

Ще одна відмінність, яку можна побачити в Inception v3, полягає в тому, щоб замінити повністю з'єднані шари в кінці простим розподілом середнього значення після останнього згорткового шару. Це різко знижує загальну кількість параметрів, що є дуже важливим досягненням, оскільки, наприклад, повноз’єднані шари мережі AlexNet містять приблизно 90% всіх параметрів моделі. Збільшення ширини та глибини мережі дозволяє Inception v3 видалити повністю з’єднані шари, при цьому зберігаючи точність результату. Inception v3 досягла 97% точності у класифікації зображень при використанні бази даних ImageNet, а також швидшої роботи порівняно з попередньо розглянутою моделлю AlexNet.

2.3 Мережева архітектура ResNet

Відомо, що мережі прямого поширення з одним шаром достатньо для представлення будь-якої функції. Але шар може бути дуже масивним і мережа стає схильною до перенавчання. Тому серед фахівців в сфері машинного навчання з'явилась тенденція розробляти архітектуру мережі в глибину. Проте, збільшення глибини мережі не можна досягати простим накладенням шарів один на одного. Нейронні мережі важко тренуються через

проблему зникнення градієнта. Оскільки градієнт повертається назад до попередніх шарів, повторне множення може зробити його нескінченно малим. Як наслідок, оскільки мережа зростає в глибину, її продуктивність може швидко знижуватися.

До появи ResNet було декілька способів вирішення проблеми зникнення градієнта, але жодного разу проблема дійсно не вирішувалась раз і назавжди.

Модель ResNet дозволяє проводити навчання глибинних мереж шляхом побудови мережі через модулі, що називаються залишковими, як показано на рисунку 2.3 [10].

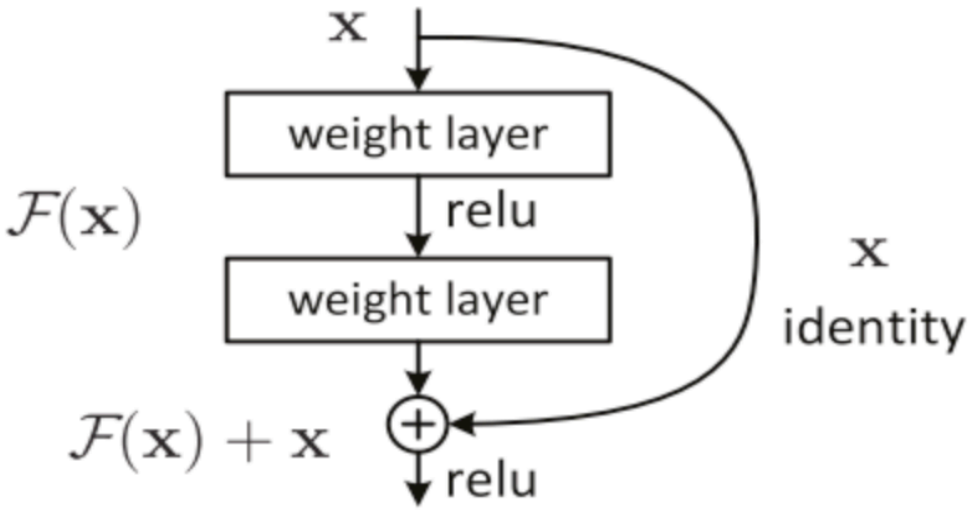

Рисунок 2.3 – Фундаментальний блок мережевої архітектури ResNet

Для того, щоб розібратись з принципами роботи залишкового блоку, уявимо мережу *A*, яка виробляє *n* помилок при навчанні. Побудуємо нову мережу *B* шляхом додавання декількох шарів до мережі *A* і задамо значення параметрів в них таким чином, щоб вони ніяк не впливали на вихідні данні, отримані з *A*. Назвемо додаткові шари як *C*. Для нової мережі залишиться та ж сама кількість помилок при навчанні, що дорівнює *n*. Отже, під час тренування мережі *B*, похибка при навчанні не повинна перевищувати ту, яка

була в *A*. Єдина причина чому так відбувається є те, що додаткові шари *C* нічого не роблять зі значеннями, які надходять з *A,* а просто копіюють їх. Це не є тривіальною задачею. Для цього модуль (блок), показаний вище, створює прямий шлях між входом і виходом до модуля, що передбачає ідентифікацію, а доданий шар *C* просто потребує вивчення ознак на вершині вже доступних вхідних даних. Оскільки *C* вивчає лише залишкові дані, весь модуль називається залишковим.

2.4 Аналіз роботи моделей AlexNet, Inception v3 та ResNet

Оскільки моделі, описані вище, мають різні архітектурні рішення та підходи до вирішення різних проблем, але всі мають однакові вимоги до вхідних даних, а також до даних для тренування (навчання мережі), пропонується розглянути та порівняти результати роботи всіх цих мережевих архітектур, використовуючи один і той же набір даних.

Для тестування роботи мереж обрана база даних зображень, що містить 100 різноманітних категорій, кожна з яких включає в себе 600 зображень однакового розміру. Ці 600 зображень поділяються на 500 навчальних та 100 тестових для кожного класу (категорії), тому загальна сума зображень, що буде використана, становить 60000. Ці 100 класів відносяться до 20 супер класів. Кожне зображення в наборі даних поставляється з двома мітками, які вказують на клас та супер клас, до яких воно належить. Вибрані категорії для навчання та тестування – це ліжко, крісло, стіл, м'який куточок, гардероб, велосипед, автобус, мотоцикл, трамвай та поїзд. Для навчання мереж потрібно використовувати хоча б декілька категорій кожного суперкласу, а використовувані супер класи – це домашні меблі та транспортні засоби. Ці категорій мають явну відмінність одна від одної, тому можна надати об’єктивну оцінку результатам роботи кожної із запропонованих для розглядання нейронних мереж.

У таблиці 2.1 показані результати роботи всіх нейронних мереж, які являють собою точність прогнозування приналежності тестового зображення до певної категорії. Оскільки кількість тестових зображень дорівнює 100, то точність вираховується як $k/100$, де k – це кількість зображень, які нейронна мережа віднесла до конкретної категорії. Щоб отримати значення у відсотках, потрібно отриману точність помножити на 100.

Таблиця 2.1 – Точність роботи нейронних мереж AlexNet, Inception v3 та ResNet

Нейронна мережа / Категорія зображення	AlexNet	Inception v3	ResNet
велосипед	21%	74.2%	55%
автобус	63.2%	84%	36.8%
мотоцикл	95%	99.2%	76.8%
поїзд	30%	95.6%	34.2%
трамвай	45.6%	83.8%	64.5%
ліжко	10%	70.8%	49.6%
крісло	89.6%	90%	57.6%
стіл	48.2%	84.6%	57.5%
м'який куточок	61%	76.4%	51%
гардероб	89%	89.4%	89.5%

Таблиця 2.1 містить результати роботи трьох мереж на тестових даних. Всі значення являють собою кількість відсотків правильного прогнозування, тобто, наприклад модель Inception v3 віднесла до дійсного класу 84 зображення автобуса зі 100, модель AlexNet мала на виході коректний результат для 63 зображень, а ResNet – для 36. Якщо подивитись на результати для решти категорій зображень, то можна побачити, що мережа Inception v3 показала кращий результат майже для всіх них.

В таблицях 2.2 та 2.3 показані точності прогнозувань для зображень

велосипеда та крісла відповідно.

Таблиця 2.2 – Точність роботи нейронних мереж AlexNet, Inception v3 та ResNet для зображення велосипеда

Нейронна мережа / Категорія зображення	AlexNet	Inception v3	ResNet
велосипед	51%	74.2%	55%
автобус	18%	0.2%	0%
мотоцикл	25%	4.4%	35%
поїзд	2%	13%	0.8%
трамвай	1%	0%	4.4%
ліжко	0%	0%	1%
крісло	0%	0.4%	0%
стіл	0%	7.6%	0.6%
м'який куточок	1%	0%	0.6%
гардероб	2%	0.2%	2.6%

Таблиця 2.3 – Точність роботи нейронних мереж AlexNet, Inception v3 та ResNet для зображення крісла

Нейронна мережа / Категорія зображення	AlexNet	Inception v3	ResNet
велосипед	0%	0%	0.2%
автобус	3%	0%	0%
мотоцикл	1%	0%	2%
поїзд	0%	0.2%	5.4%

Нейронна мережа / Категорія зображення	AlexNet	Inception v3	ResNet
трамвай	0%	0%	0.6%
ліжко	0%	4%	7.4%
крісло	90%	92.6%	57.6%
стіл	0%	2.8%	0%
м'який куточок	1%	0%	21%
гардероб	5%	0.4%	5.8%

Проаналізувавши результати, надані в таблицях 2.2 та 2.3, можна помітити, що і в конкретних тестових випадках мережева архітектура Inception v3 поводиться краще за AlexNet та ResNet. Таке спостереження дає змогу обрати саме цю моделі для вирішення поставленої задачі, оскільки, не проводивши ніяких хоча б малих досліджень, важко визначитись з тим, яку саме мережу використовувати при розробці власної, тобто на основі якої моделі здійснювати алгоритм перенавчання.

Ще одним критерієм для порівняння попередньо навчених моделей є кількість параметрів, які були отримані в ході їхнього тренування і які використовуються в процесі їхнього перенавчання. Зрозуміло, що час, який буде затрачений на тренування нової мережі (наприклад, на основі однієї з розглянутих вище) напряму залежить від цієї кількості, що може буде вирішальним моментом у випадку, коли є обмеження, пов’язані з обчислювальними ресурсами, або коли цього вимагає будь-яка конкретна задача, де класифікація зображень є лише частиною обробки даних. Також на час впливає число шарів, які лежать в основі архітектурної реалізації. Тому далі пропонується порівняти моделі за цими критеріями.

Кількість шарів в моделях, що розглядаються, приблизно дорівнює 150 і може відрізнятись на декілька шарів в певних реалізаціях, що є несуттєвою

відмінністю від середнього значення. Тобто всі три мережі мають схожу ситуацію. А ось число параметрів відрізняється в кожній з них. Наприклад, мережева архітектура AlexNet має кількість параметрів, що дорівнює 60 мільйонам, ResNet – 25 мільйонів, і найменший показник у Inception v3 – 23 мільйони.

В останні роки фахівці в сфері машинного навчання, що займаються розробками нових моделей для вирішення задач, пов'язаних з комп'ютерним зором, намагаються скоротити число параметрів нейронної мережі, при цьому зберігаючи її продуктивність і навіть навпаки, зменшуючи відсоток помилок результатів. Число помилок на виході конволюційних нейронних мереж спадає з роками, як показано на наступному графіку (рис. 2.4).

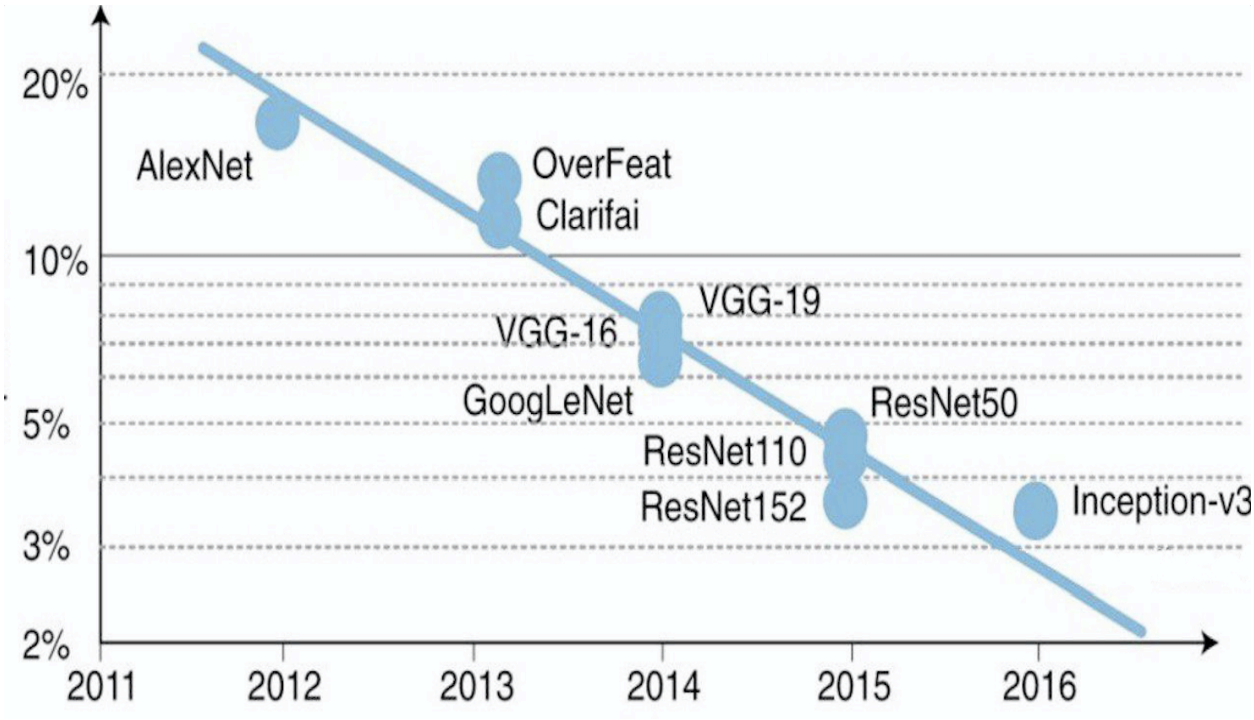

Рисунок 2.4 – Графік спадання відсотку помилок в моделях, навчених на базі даних ImageNet

На вищенаведеному графіку можна побачити, що модель Inception v3 має найнижчі показники (малий відсоток помилок при класифікації

зображень), що висуває її вперед порівняно з рештою розробок.

Результати аналізу по всім запропонованим критеріям показали, що нейронна мережа Inception v3 має не складнішу архітектуру, ніж ті, що лежать в основі AlexNet та ResNet, має найменшу кількість параметрів, які будуть використанні під час перенавчання, а також найнижчу кількість помилкових результатів на виході. До того ж кількість часу, необхідного для тренування, є не більшим за той, що витрачається у випадку використання двох інших моделей. Тому Inception v3 є оптимальним вибором для вирішення задачі класифікації зображень.

2.5 Метод перенавчання з використанням Inception v3

Сучасні нейронні мережі, які використовуються для розпізнавання зображень, мають мільйони параметрів, як було зазначено вище. Навчання їх з нуля потребує багато навчальних даних з відповідними мітками для класифікації та великої обчислювальної потужності (сотні годин графічного процесора або більше). Передача навчання (метод перенавчання) – це техніка, яка значно полегшує процес розробки мережі, беручи частину моделі, яка вже була підготовлена на відповідних даних, та використовуючи її в новій моделі з усіма раніше обчисленими параметрами. Далі буде розглянутий цей метод з використанням потужного класифікатора зображень, навченого на базі даних ImageNet. Хоча для кожної нової задачі класифікації зображень, виходячи з її унікальності, частіше за все підготовлюють (навчають) нову модель з нуля, але метод перенавчання може бути ефективним та оптимальним рішенням для багатьох додатків, оскільки він вимагає невеликої кількісті навчальних даних (тисячі, а не мільйони зображень з проставленими мітками), а також час його виконання може складати всього тридцять хвилин на комп’ютері без графічного процесора.

2.5.1 Підготовка даних для навчання

У вирішенні будь-якої задачі класифікації зображень першим та самим важливим кроком є підготовка даних для навчання. Оскільки пропонується взяти за базу та використовувати модель Inception v3, кількість необхідних даних вираховується вже не в мільйонах, а в сотнях або тисячах. Для того, щоб підготовлена мережа видавала задовільні результати (а це більше 90% точності розпізнавання), потрібно підготувати щонайменше сто фотографій кожного виду об'єкта, який буде розпізнаватись. Чим більше таких фотографій, тим краще буде точність навченої моделі. Також необхідно переконатись, що вони охоплюють всі можливі випадки, які будуть дійсно подаватись на вхід класифікатора. Наприклад, якщо всі тестові знімки зроблені в приміщенні на фоні порожньої стіни, а користувач намагається розпізнати об'єкти на вулиці, то, напевно, задовільних результатів на виході не буде отримано. Ще одна проблема, яка полягає у недостатній кількості підготовлених даних, описується наступною ситуацією. Наприклад, якщо зображення одного з видів об'єкта має блакитний фон, а іншого – зелений, тоді модель прогнозуватиме приналежність зображення до того чи іншого класу, базуючись на фоновому кольорі, а не на об'єктах, що містяться на ньому. Щоб уникнути цього, необхідно мати якнайбільше різних варіантів зображень одного і того ж об’єкта класифікації. Також слід правильно визначити категорії, які будуть використовуватись для класифікації. Може варто розділити великі категорії, які охоплюють багато різних фізичних форм у менші, більш конкретні. Наприклад, замість “транспортний засіб” можна використовувати “машина”, “мотоцикл” та “вантажівка”.

Для того, щоб скрипт, який запускає навчання моделі, почав використовувати підготовлені дані, структура зображень повинна бути наступною (як показано на рисунку 2.5): всі зображення однієї категорії необхідно помістити в одну директорію, яка має назву класу цих даних, і всі

такі директорії помістити в одну загальну, назва якої може бути довільною. Назву (шлях до цієї кореневої папки) необхідно передати в якості аргумента *image_dir* під час запуску скрипта.

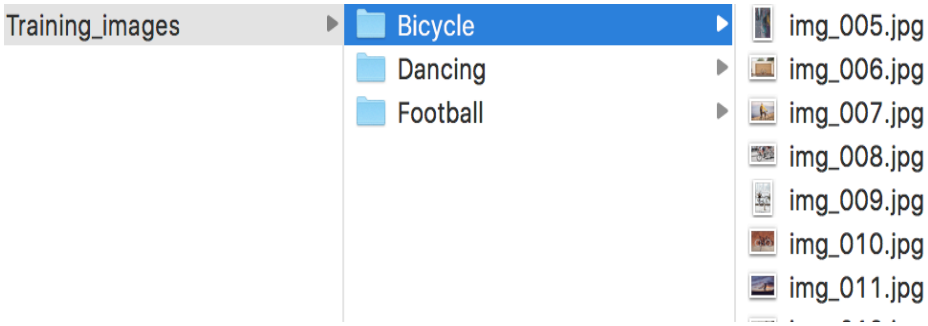

Рисунок 2.5 – Ієрархічна структура даних для навчання

Наступним етапом після підготовки даних для перенавчання готової моделі Inception v3 є налаштування параметрів для запуску скрипта, за допомогою якого мережа тренується на цих даних.

2.5.2 Параметри процесу навчання нейронної мережі

Якщо необхідні для тренування моделі зображення підготовлені, можна приступати до задання значень параметрів, що використовуються в процесі перенавчання. Перший важливий аргумент запуску скрипта, значення якого можна змінити, – це *how_many_training_steps*. Це значення за замовчуванням становить 4000, але якщо збільшити його до 8000, нейронна мережа буде тренуватися вдвічі довше. Збільшення точності сповільнюється із зростанням числа кроків тренування і в якийсь момент повністю зупиняється (або навіть починає спадати через перенасичення мережі), але потрібно експериментувати, дивлячись на результати навчання, щоб дізнатися, що найкраще підходить для конкретної моделі.

Звичайним способом покращення результатів навчання зображень є деформація, обрізка або освітлення навчальних даних випадковими

способами. Це надає перевагу, яка полягає у збільшенні ефективного розміру навчальних даних завдяки всім можливим варіаціям тих самих зображень і, як правило, допомагає мережі навчитися оброблювати всі можливі деформації або варіації, які будуть виникати під час використання класифікатора. Найбільшим недоліком уможливлення цих спотворень є те, що кешування вузьких місць більше не є корисним, оскільки вхідні зображення ніколи не мають сталої форми. Це означає, що процес тренування триває набагато довше (декілька годин), тому рекомендується спробувати це як спосіб полірування моделі лише після того, як досягнуто результатів, які близькі до очікуваних, але з трохи меншою точністю.

Увімкнути ці опції можна, задаючи значення таких аргументів, як *random_crop*, *random_scale* і *random_brightness*. Всі вони вказуються у відсотках. Параметри контролюють, наскільки велика частина кожного з перетворень застосовується до кожного зображення. Рекомендують починати зі значень 5 або 10 для кожного з них, а потім експериментувати, щоб побачити, що з них допомагає покращити результати, а що навпаки. Аргумент *flip_left_right* буде випадковим чином відображати половину зображень горизонтально, що має сенс, якщо ці інверсії, ймовірно, відбудуться під час використання моделі.

Є декілька інших параметрів, які також можна спробувати налаштувати, щоб побачити, чи допоможуть вони у збільшенні точності результатів. Параметр *learning_rate* контролює, наскільки часто оновлення мають надходити до останнього шару під час тренувань. Інтуїтивно зрозуміло, що при меншому значенні цього аргумента навчання займає більше часу, але це може збільшити загальну точність розпізнавання. Однак це не завжди так, тому потрібно також експериментувати, щоб дізнатись чи допоможе це налаштування в роботі мережі. Параметр *train_batch_size* контролює кількість зображень, що оброблюються під час кожного навчального кроку. Далі буде показані та проаналізовані результати навчання

нейронної мережі в залежності від заданих параметрів, розглянутих вище.

Для того, щоб вміти аналізувати результати виведенні на екран під час та після навчання моделі, розглянемо що ж саме виконує скрипт і на основі яких даних вираховує точність роботи мережі.

Першим, що виконує програма, коли отримує в якості параметра шлях до директорії з усіма даними, необхідними для навчання моделі, поділяє ці дані на три різні набори: набір для тренування, набір для перевірки, що використовується під час навчання, та тестовий. Найбільшою зазвичай є перша вибірка, яка під час тренувань передає всі зображення у мережу, результати обробки яких використовуються для оновлення ваг моделі. Можна задатися питанням, чому ж не використовувати всі зображення для навчання. Велика проблема, яку можна зустріти досить часто в сфері машинного навчання, полягає в тому, що модель може просто запам'ятовувати невідповідні деталі навчальних зображень, щоб надати правильні відповіді. Наприклад, уявимо собі мережу, яка запам'ятовує шаблон кожної фотографії, і використовує його для створення відповідності між класами та об'єктами. В такому випадку мережа впорається з класифікацією зображень, які вона бачила під час тренування, але не зможе правильно віднести нове для неї зображення до того чи іншого класу через те, що не були вивчені загальні характеристики об'єктів, а просто запам’ятовувались несуттєві деталі тренувальних зображень.

Ця проблема має назву “перенавчання”, що походить від англійського “overfitting”, і щоб її уникнути, деякі дані зберігаються поза навчальним процесом, щоб модель не змогла запам'ятати їх. Потім ці зображення використовуються як перевірка, щоб переконатись, що перенавчання не відбувається, оскільки якщо ми бачимо досить високу точність, це підкреслює, що мережа не перенавчена. Звичайним розподілом даних є наступним: 80% зображень потрапляють в основний набір (тренувальний), 10% – у перевірочний, щоб часто виконувати процедуру валідації під час

тренувань, і 10% – у тестовий набір, що використовується для оцінки прогнозування реального часу. Ці співвідношення можна регулювати за допомогою параметрів *testing_percentage* та *validation_percentage*. Загалом, ці значення залишаються за умовчанням, оскільки, як правило, не знайдеться ніяких переваг для навчання при їх налаштуванні.

Скрипт використовує назви файлів зображень (а не повністю випадкову функцію), щоб розділити зображення між навчальним, перевіряючим та тестовим наборами. Це робиться для того, щоб зображення не переходили між наборами для тренувань та тестування при різних запусках, оскільки це може бути проблемою, якщо зображення, які були використані для навчання моделі, згодом потрапляють до набору перевірки.

Точність перевірки змінюється з кожною ітерацією. Значна частина цієї зміни випливає з того факту, що для кожного вимірювання точності роботи мережі вибирається випадкова підмножина набору перевірки. Коливання цих значень можуть бути значно зменшені, за рахунок деякого збільшення часу навчання, встановивши *validation_batch_size = -1*, який говорить мережі використовувати весь набір перевірки для кожного обчислення точності.

Після завершення тренування можливо виявити неправильно класифіковані зображення серед тестового набору. Це можна зробити, додавши прапорець *print_misclassified_test_images*. Це може допомогти зрозуміти, які типи зображень були найбільш заплутаними для моделі, і які категорії було найбільш важко відрізнити. Наприклад, можна виявити, що певний підтип певної категорії або якийсь незвичайний кут фотографії особливо важко визначити, що говорить про те, що слід додати нові навчальні зображення цього підтипу. Нерідко виявлення неправильно класифікованих зображень може також вказувати на помилки в наборі вхідних даних, наприклад, неправильно позначені, низькоякісні або неоднозначні зображення. Проте загалом слід уникати виправлення окремих помилок у тестовому наборі, оскільки вони, ймовірно, лише відображають

більш загальні проблеми у навчальному наборі.

Визначивши як підготовлювати зображення і яку структуру вони повинні мати, а також розглянувши роботу та параметри скрипта, що виконує перенавчання, можна починати створювати власну модель. Але, оскільки модель Inception v3 призначена класифікувати зображення за однією міткою (класом), необхідно модифікувати алгоритм навчання таким чином, щоб можливо було виконувати багатокластерну класифікацію.

2.6 Модифікований метод для багатокластерної класифікації

Для того, щоб нейронна мережа під час тренування змогла визначити класи об’єктів, які містяться на зображенні, необхідно змінити підхід підготовки даних для навчання. Для цього потрібно:

1) помістити усі навчальні зображення в одну директорію з довільною назвою, при цьому видаливши всі дубльовані зображення, які можуть впливати на точність тестування та перевірки;
2) підготувати файл з назвами класів об’єктів, які знаходяться на кожному зображенні. Назва файлу повинна бути у форматі *<назва_файлу_із_зображенням.jpg>.txt*, наприклад, якщо файл із зображенням має назву *circle.jpg*, то файл з мітками (класами) повинен називатись *circle.jpg.txt*. Кожна назва класу у такому файлі повинна починатись з нової строки. Всі файли потрібно помістити у директорію *image_labels_dir,* шлях до якої можна змінити, задавши параметр *IMAGE_LABELS_DIR*.

За замовчуванням модель Inception v3 використовує структуру директорій із зображеннями для навчання, щоб отримати список категорій для класифікації. Оскільки було вирішено класти всі навчальні зображення всередину однієї директорії, потрібно перерахувати всі можливі класи у зовнішньому файлі. Тому наступним кроком є створення файлу *labels.txt* у

корені проекту і заповнення його усіма можливими мітками (назвами класів). Кожна назва, як і у файлі з назвами категорій для кожного окремого зображення, повинна починатись з нової строки.

Головна функція скрипта, що виконує процес перенавчання мережі, спочатку завантажує структуру каталогу, що містить зображення для кожного класу в окремій директорії, і створює на основі цієї структури набори для перевірки, тестування та навчання для кожного класу. Оскільки було вирішено зберігати всі навчальні зображення всередині однієї папки, масив *image_lists* буде містити лише один елемент, тобто цю папку. Всі три набори зображень будуть створені з цього одного елемента. Тепер необхідно лише правильно отримати число та назви всіх можливих класів зображень, для чого і створювався файл *labels.txt.*

На початку обробки кожного зображення створюється так зване “вузьке місце”, яке являє собою вектор з довжиною, що дорівнює числу класів. Кожний елемент цього вектору відповідає якомусь класу об’єктів, значення якого може бути 0 або 1, де 1 означає, що на зображенні знаходиться об’єкт цього класу, а 0 – навпаки. Спочатку всі значення цього вектору дорівнюють 0. Потім заноситься 1 на позицію конкретного класу, назва якого дізнається з назви директорії, з якої було отримане зображення. У випадку багатокластерної класифікації ця процедура дещо змінюється. Потрібно отримати назви всіх класів, до яких належить вхідне зображення, та занести 1 до вектору у всіх відповідних позиціях. Назви класів отримуються з текстового файлу, що містить описані категорії. Для кожного зображення при підготовці навчальних даних має бути створений окремий такий файл.

В основі методу оцінювання прогнозувань алгоритму лежить функція активації (або передаточна функція) *softmax*, яка перетворює всі значення вектору числових прогнозів таким чином, щоб вони знаходились в діапазоні від 0 до 1, а їх сума дорівнювала 1. Ця функція необхідна у випадку класифікації за однією категорією. Але для багатокластерної класифікації

вона не підходить, адже результат роботи мережі повинен складати ймовірності приналежності зображення до кожного з підкласів. Наприклад, можна сказати, що зображення належить до класу “транспортний засіб” з ймовірністю 0.85, а також до класу “мотоцикл” з ймовірністю 0.67. Сума цих значень більше 1. Тому для досягнення необхідного результату пропонується модифікувати метод оцінювання прогнозувань, використовуючи замість *softmax* іншу функцію активації.

2.7 Функції активації в згортковій нейронній мережі

Функції активації (передавальні функції) є надзвичайно важливою частиною штучних нейронних мереж. Вони в основному вирішують, чи потрібно активувати нейрон, тобто чи інформація, яку отримує нейрон, є релевантною для вхідних даних, чи її слід ігнорувати. Існує два типи функцій активації: лінійні та нелінійні. Лінійні можуть бути представлені функцією $f(x) = x$ і вираховують вихідне значення, яке має прямо пропорційну залежність від вхідного. Нелінійні функції використовуються для відокремлення даних, які не є лінійно відокремленими, і є найбільш часто використовуваними функціями активації в конволюційних нейронних мережах. Нелінійне рівняння допомагає належним чином вирахувати вихідне значення. Кілька прикладів різних типів нелінійних функцій активації – це функція сигмоїду (sigmoid), тангенса (tanh) та лінійний блок випрямлення (ReLU – з англ. Rectified Linear Unit) [11]. Саме їх далі пропонується розглянути, оскільки вони є найпоширенішими передавальними функціями в сучасних розробках.

Нелінійні активаційні функції дозволяють нейронним мережам наближати досить складні функції. Без нелінійності, представленою такими функціями, декілька шарів нейронної мережі еквівалентні однорідній нейронній мережі. Розглянемо простий приклад, щоб зрозуміти, чому без

нелінійності неможливо наблизити навіть таку просту функцію, як XOR. На рисунку 2.6 графічно показано результати роботи цієї функції.

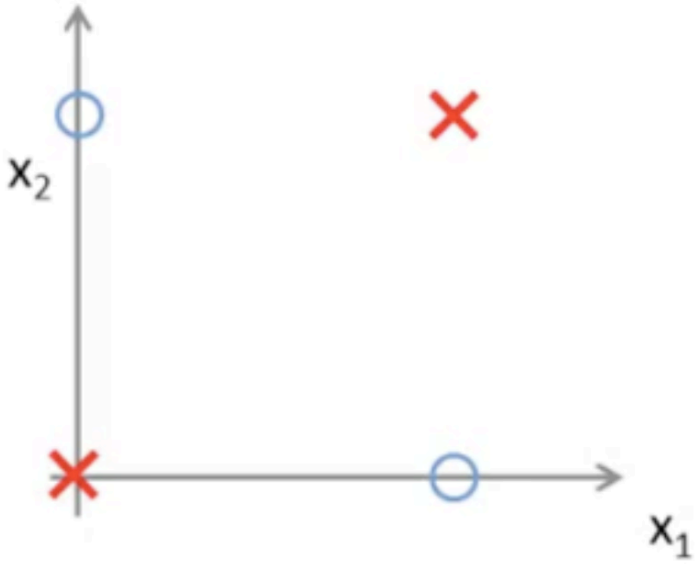

Рисунок 2.6 – Графічне представлення нелінійності відокремлення даних при використанні функції XOR

Допустимо у наборі даних є два класи, представлені хрестиком та колом. Коли два параметра X1 та X2 однакові, клас на виході – хрестик, інакше – коло. Два хрестика мають вихідне значення 0 для вхідних значень (0,0) та (1,1), а два кола мають на виході 1 для вхідних значень (0,1) і (1,0).

З наведеної вище картини можна побачити, що точки на графіку не є лінійно відокремленими. Іншими словами, неможливо провести пряму лінію, щоб відокремити кола та хрести один від одного. Отже, потрібне рішення для проведення нелінійної границі між точками.

Функція активації також є відповідальною за перетворення результуючого значення нейронної мережі в таке, що буде знаходитись в певних межах. Вихідні дані нейрона, що зображений на рисунку 2.7, можуть бути представлені дуже великими числами, які обраховуються за формулою:

$$\sum_{i}^{n} w_i x_i + b.$$

де x_i – це значення вхідного вектору, що множиться на вагу з'єднання w_i, після чого до суми всіх таких добутків додається зміщення b.

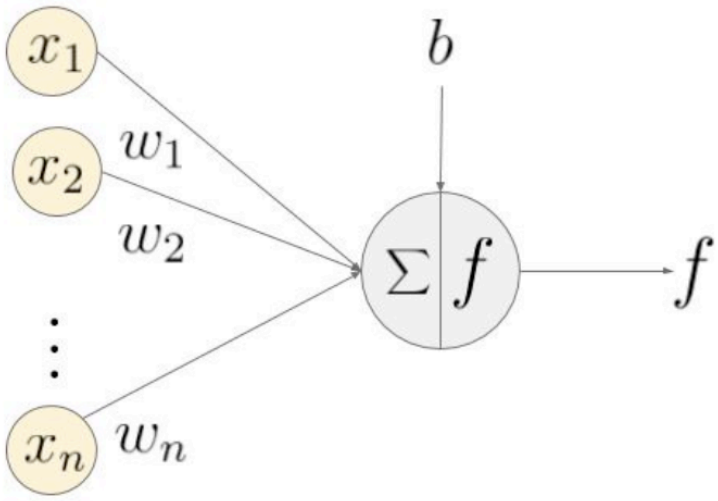

Рисунок 2.7 – Схема нейрона згорткової нейронної мережі

Цей вихід, коли подається до наступного нейрона шару без модифікації, може бути перетворений на ще більше число, що робить процес обчислень досить складним. Одне із завдань функції активації, яка на рисунку 2.7 позначається як *f*, полягає в тому, щоб перетворити вихід нейрона на щось обмежене (наприклад, значення від 0 до 1).

Визначивши призначення нелінійної функції активації, можна розглянути декілька їх видів та порівняти для обрання та подальшого використання у розроблюваній нейронній мережі. До сих пір найчастіше застосовують сигмоїдну функцію, яка має вже відомі недоліки при використанні її в прихованих шарах згорткової нейронної мережі.

2.7.1 Функція активації Sigmoid

Функція сигмоїду, графік якої зображений на рисунку 2.8, також відома

як логістична функція активації. Вона приймає число, отримане в результаті обчислень, розглянутих вище, і перетворює його таким чином, щоб воно знаходилось в діапазоні від 0 до 1.

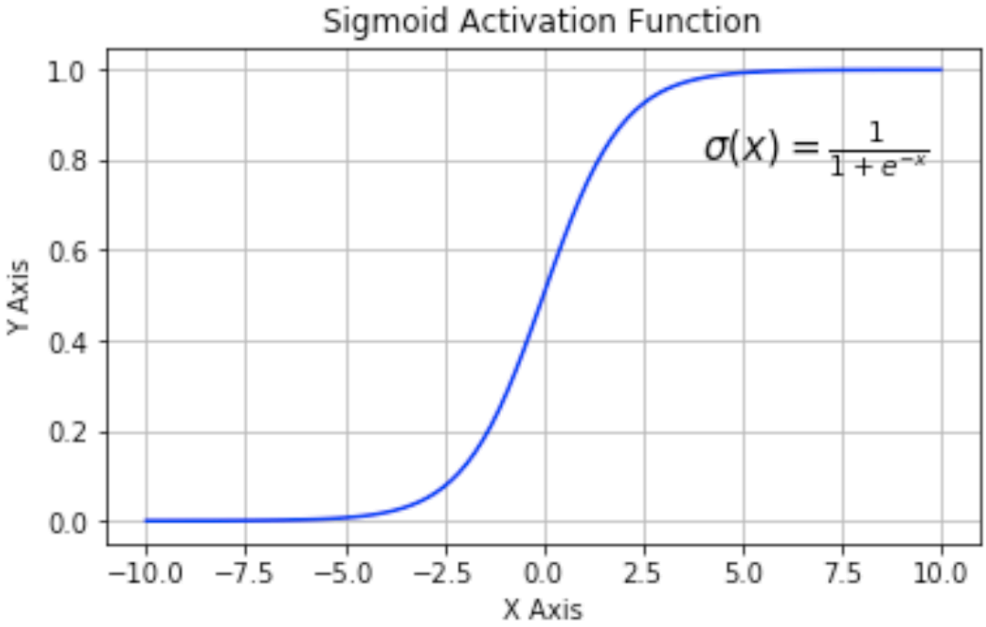

Рисунок 2.8 – Графік сигмоїдної функції активації

Найчастіше ця функція використовується у вихідному шарі конволюційної нейронної мережі, задача якого полягає у прогнозуванні ймовірності приналежності вхідних даних (зображення) до певного класу. Вона перетворює великі від'ємні числа на 0 і великі додатні числа на 1. Математично представляється як

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \cdot$$

На графіку, зображеному на рисунку 2.8, можна побачити, що значення функції сигмоїду є незмінними при великих від’ємних або додатних числах на вході. Іншими словами, градієнт сигмоїду дорівнює 0 в таких випадках. Поняття градієнта використовується в контексті навчання нейронної мережі, коли застосовується алгоритм зворотного поширення для оновлення ваг нейронів. Градієнт вираховується за допомогою взяття похідної функції

втрат, яка обчислює різницю між реальним вихідним значенням нейронної мережі та очікуваним. При зворотному поширенні через мережу з сигмоїдною активацією градієнти в нейронах, вихід яких близький до 0 або 1, дорівнює майже 0. Ці нейрони називаються насиченими. Таким чином, ваги в цих нейронах не оновлюються. Мало того, ваги нейронів, з'єднаних з такими нейронами, також повільно оновлюються. Ця проблема має назву зникаючого градієнту.

Також функція Sigmoid має проблему, яка полягає у тому, що вихідні значення нецентровані відносно нуля. Цю проблему вирішує наступна функція активації – функція тангенса.

2.7.2 Функція активації Tanh

Функція тангенса також відома як гіперболічна функція активації дотичної. Графік зображений на рисунку 2.9. Подібно до сигмоїдної, tanh також приймає довільне число, але перетворює його таким чином, щоб воно знаходилось в діапазоні між -1 і 1. На відміну від вихідних значень сигмоподібних функцій, значення tanh центровані відносно нуля, оскільки область значень знаходиться між -1 та 1. На практиці функція тангенса краще, ніж сигмоїдна, оскільки вона менш складна в обчисленні. Від’ємні вхідні дані розглядаються як від’ємні, нульові значення залишаються нульовими, а додатні перетворюються на додатні.

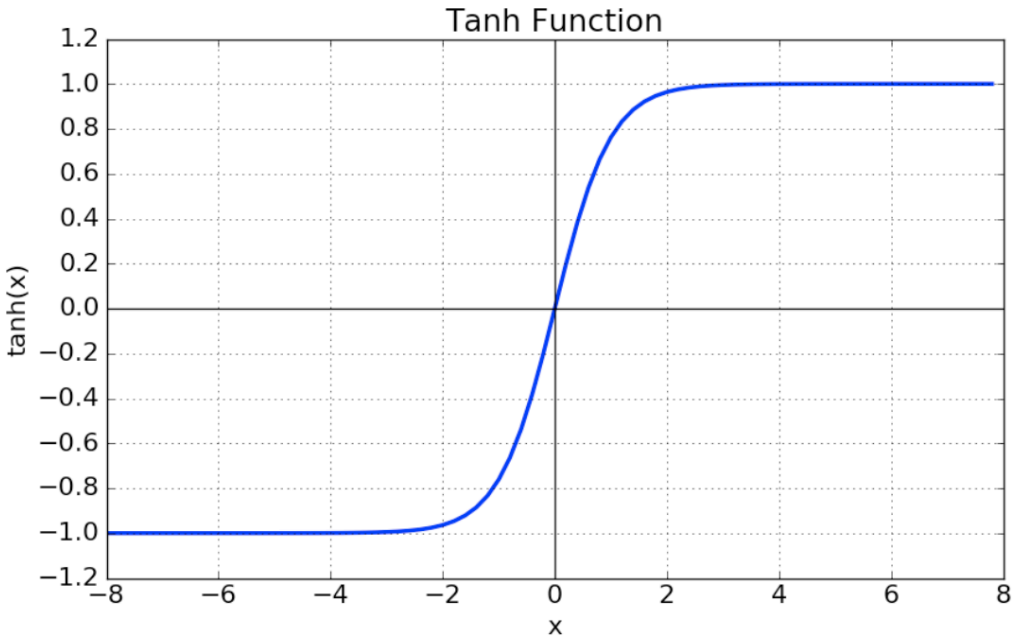

Рисунок 2.9 – Графік функції активації tanh

Єдиним недоліком цієї функції є те, що залишається проблема зникаючого градієнта. Саме тому вже декілька років замість сигмоїдної та функції тангенса в прихованих шарах згорткової нейронної мережі використовується інша функція активації, яка буде розглянута далі.

2.7.3 Лінійний блок випрямлення ReLU

Лінійний блок випрямлення наполовину випрямлений знизу, як видно на рисунку 2.10. Математично це описується простим виразом:

$F(x) = \max(0, x)$.

Це означає, що коли вхідне значення $x < 0$, вихідне буде дорівнювати 0, а якщо $x > 0$, то на виході x.

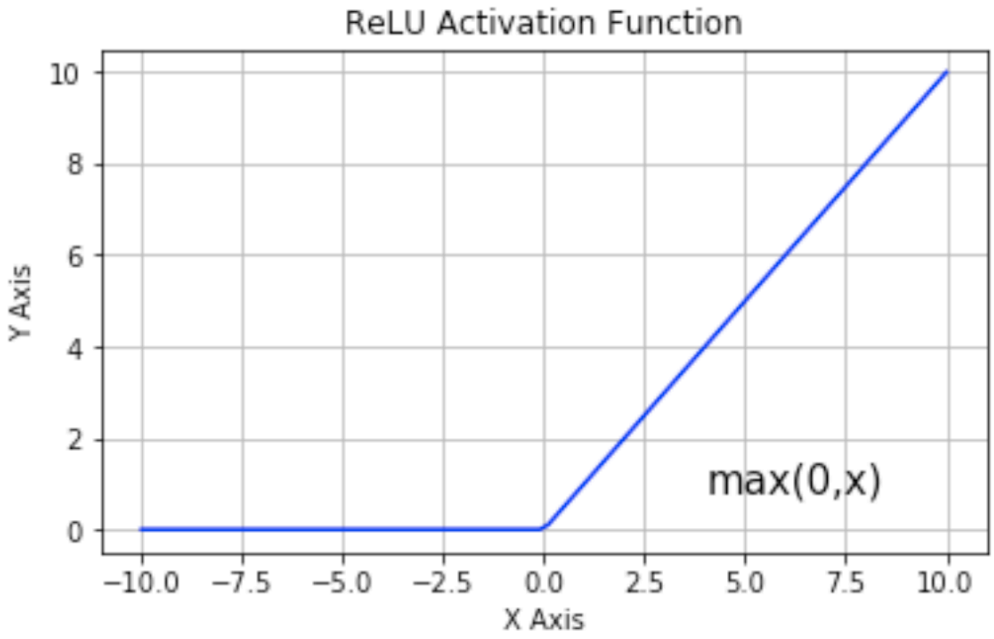

Рисунок 2.10 – Графік функції активації ReLU

Ця функція активації робить навчання мережі набагато швидшим. Вона не призводить до насичення нейронів, а це означає, що вона є стійкою до проблеми зникаючого градієнта принаймні при $x > 0$, тому нейрони зворотно не поширюють всі нулі принаймні у половині випадків. Обчислення цієї функції є нескладним, оскільки воно реалізується за допомогою простого порогу. Але є декілька недоліків використання функції ReLU:

1) вихідні значення не центровані відносно нуля подібно тому, як при використанні функції sigmoid;
2) проблема зникання градієнта під час зворотного поширення залишається, але тільки при $x < 0$, тобто ваги не оновлюються і мережа не навчається.

Отже, розглянувши та проаналізувавши три різні нелінійні функції активації, можна зробити вибір однієї з них. Слід нагадати, що було вирішено замінити функцію softmax, яка використовується в останньому шарі згорткової нейронної мережі Inception v3 для прогнозування ймовірності приналежності зображення до певного класу, оскільки вона, як і

всі розглянуті вище функції, перетворює вихідне значення нейрона на значення в діапазоні від 0 до 1, сума яких дорівнює 1. Це підходить для класифікації зображень за однією міткою, але для багатокластерної класифікації необхідно обрати іншу функцію активації. В останні роки функції sigmoid та tanh не використовуються в прихованих шарах конволюційної нейронної мережі через проблему зникання градієнта при навчанні, яку частково вирішує лінійний блок випрямлення. Тому функція активації ReLU лежить в основі моделі Inception v3. Але необхідно замінити функцію, яка розташована на виході мережі і на процес навчання не впливає. Її задача полягає у видачі результату, який являє собою сукупність ймовірностей приналежності зображення до категорії. Тобто вхідними даними для цієї функції є вектор, розмірність якого дорівнює кількості класів, із значеннями, що відповідають кожній категорії, а вихідними – вектор такої ж розмірності з іншими, перетвореними значеннями (ймовірностями). При перетворенні значень вихідного вектору необхідно зберігати їх співвідношення, тобто чим більшим було значення, тим ближчим до 1 воно має буде на виході. А оскільки ймовірність не може бути від’ємною, вирішено використовувати сигмоїдну функцію активації на виході нейронної мережі.

Останнім, що залишилось розглянути, є засоби для запобігання такому ефекту як перенавчання, сутність якого була згадана при описі методики розділення даних для навчання мережі на три набори: тренувальний, тестовий та перевірочний. Але з перенавчанням моделі можна зіткнутись з різних причин, однією з яких є надмірна зв’язність окремих одиниць нейронної мережі.

2.8 Засоби для запобігання перенавчання нейронної мережі

Дізнатись про те, що розроблювана модель для класифікації

перенавчена або навпаки недостатньо навчена, можна за допомогою різних спостережень. Наприклад, модель недостатньо натренована, якщо точність, отримана при використанні перевірочного набору даних, перевищує точність на наборі для тренувань. Окрім того, якщо остаточна модель має на виході незадовільні результати, це також говорить про наявність цього ефекту. Наприклад, використання лінійної моделі для розпізнавання зображень, як правило, призведе до створення моделі, що не відповідає вимогам. Також недостатньо навчена модель може бути отримана в результаті надмірного використання метода виключення (з англ. “dropout”). Відкидання випадковим чином деяких нейронів або встановлення результату його функції активації в нуль під час тренувального процесу допомагає уникнути іншого згаданого ефекту – перенавчання. Метод виключення не застосовується під час прогнозування на перевірочному або тестовому наборах. Якщо модель перенасичена, можна почати додавати шари виключення, починаючи з невеликої частини мережі.

Надмірне навчання, проблема якого графічно представлена на рисунку 2.11, відбувається, коли модель добре працює, тобто показує високу точність, на наборі даних, який використовується для тренування. Після цього стає складно узагальнити модель так, щоб вона так само опрацьовувала нові приклади, які не були включені до тренувального набору. Це говорить про те, що модель розпізнає певні зображення з цього набору замість загальних властивостей. Точність навчання в цьому випадку буде вищою, ніж точність на перевірочному та тестовому наборах.

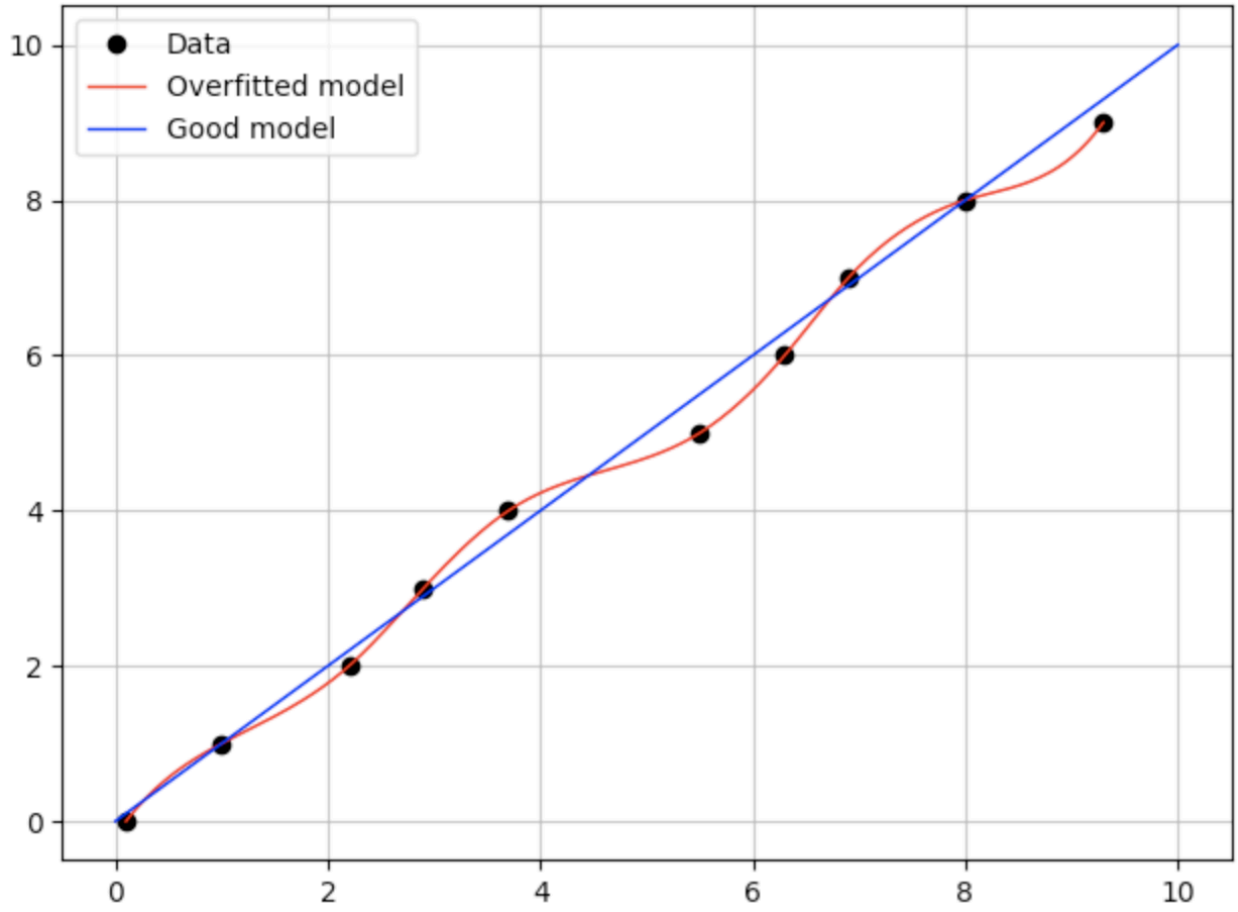

Рисунок 2.11 – Графічне представлення проблеми перенавчання нейронної мережі

Дані на цьому графіку представлені жирними точками. Крива лінія – це перенавчена модель, яка, як можна побачити, натренована таким чином, що розпізнає надлишкові дані, тобто шум. Пряма лінія – це модель без наявності цього ефекту.

Для того, щоб запобігти перенавчанню розроблюваної нейронної мережі, застосовують декілька методів та прийомів [12]:

- використання більшого об’єму даних для навчання;
- збільшення числа зображень за рахунок таких операцій, як перевертання, масштабування та накладення різноманітних фільтрів;

- додавання регуляризації (в основному використовують метод виключення).

Перший спосіб, звичайно, полягає в тому, щоб зібрати більше даних для навчання, другий – у їх модифікації. Однак у більшості випадків є деякі обмеження, які не дозволяють цього зробити, наприклад, вимоги до часу на підготовку та навчання або обмежена кількість даних. А другий спосіб, зазвичай, застосований з самого початку. Тому додавання регуляризації є дуже важливим для вирішення проблеми перенавчання нейронної мережі. У глибинному навчанні найчастіше використовується метод виключення, наприклад, він застосовується в моделі AlexNet. Шар виключення можна помістити в будь-яке місце в нейронній мережі (після будь-якого з прихованих шарів, після повністю з'єднаного та навіть вхідного), але доцільно це робити поступово, підключаючи його спочатку після передостаннього шару мережі та дивлячись на те, як це впливає на результати роботи моделі. Адже надмірне використання цього шару може призвести до того, що мережа буде пропускати якісь важливі ознаки, якщо, наприклад, виключення буде відбуватись одразу після вхідного шару. Тому вирішено використовувати цей метод для запобігання перенавчанню нейронної мережі, але тільки в кінці, безпосередньо перед видачою результату.

Для програмної реалізації всього вищесказаного використовується потужна бібліотека TensorFlow.

2.9 Засоби для аналізу процесу навчання нейронної мережі

Дуже часто для вирішення задач машинного навчання застосовують мову програмування Python. Однією з причин вибору саме цієї мови є велика кількість додатків, розроблених для різноманітних математичних обчислень і не тільки. Для створення архітектури та роботи з нейронними мережами

створена бібліотека TensorFlow, яку вирішено використовувати для реалізації запропонованого методу перенавчання моделі Inception v3. TensorFlow – це бібліотека з відкритим кодом, створена для виконання складних обчислень, що надає змогу прискорити процес машинного навчання та робить його швидшим. Додатки, розроблені за допомогою цієї бібліотеки, можуть виконуватись на будь-яких пристроях: локальному комп’ютері, кластері у хмарі, iOS та Android смартфонах, графічних процесорах тощо.

TensorFlow дозволяє розробникам створювати графічно-структуровані моделі даних, що описують, як дані рухаються через граф або серію оброблюваних вузлів. Кожен вузол виконує математичну операцію, а кожне з'єднання між ними представляє собою багатовимірний масив даних (або тензор).

Великою перевагою бібліотеки, що полегшує роботу з задачами машинного навчання, є рівень абстракції, який вона забезпечує. Замість того, щоб мати справу з деталями алгоритмів, що застосовуються, або знаходженням належних способів організації функцій, розробник може зосередити свою увагу на загальній логіці програми. TensorFlow відповідає за деталі самостійно.

TensorFlow пропонує додаткові засоби для налагодження та аналізу розроблюваних програм. Наприклад, є можливість застосовувати особливий режим виконання, який дозволяє оцінювати та змінювати роботу кожної операції графа окремо, замість того, щоб будувати весь граф та аналізувати його. Комплект візуалізації TensorBoard дозволяє перевірити та проаналізувати роботу нейронної мережі за допомогою інтерактивної web-орієнтованої інформаційної панелі.

Візуалізація процесу навчання нейронної мережі є дуже зручним засобом перевірки його коректності, оскільки є можливість наглядно побачити зміни в роботі моделі, пов’язані із зміненням одного або декількох початкових параметрів. Вирішено використовувати TensorBoard, що дозволяє

переглядати будь-які графіки моделі, збільшуючи масштаб для виявлення деталей.

За допомогою TensorBoard можна будувати графіки показників, таких як втрата та точність під час тренувань, гістограми про те, як тензор змінюється з часом, показувати додаткові дані, такі як зображення, які проходять через мережу під час навчання, збирати метадані про виконання, наприклад, загальне використання пам'яті тощо.

TensorBoard працює, читаючи файли подій, які містять зведені дані, які можна генерувати під час запуску TensorFlow програм. Для того, щоб під час виконання скрипта, який виконує перенавчання існуючої моделі, створювались файли із цими даними, необхідно встановити значення параметра *summaries_dir* та вказати шлях до директорії, що буде містити ці файли.

Щоб запустити TensorBoard, необхідно виконати наступну команду: *tensorboard --logdir=path_to_log_directory,* де *path_to_log_directory* – це шлях до директорії із серіалізованими даними, який був переданий в якості параметра *summaries_dir* під час запуску перенавчання. Якщо цей каталог містить підкаталоги, які містять серіалізовані дані окремих запусків, то TensorBoard опрацьовує кожний з них. Після виконання вищенаведеної команди, необхідно перейти до ресурсу за адресою localhost:6006, щоб переглянути панель з усіма згенерованими даними. Наприклад, найчастіше застосовуваними показниками, що надають змогу проаналізувати процес навчання мережі, є точність роботи моделі та значення крос-ентропії на різних етапах, приклади графіків яких наведені на рисунках 2.12 та 2.13 відповідно. Крос-ентропія є функцією втрат для згорткової нейронної мережі.

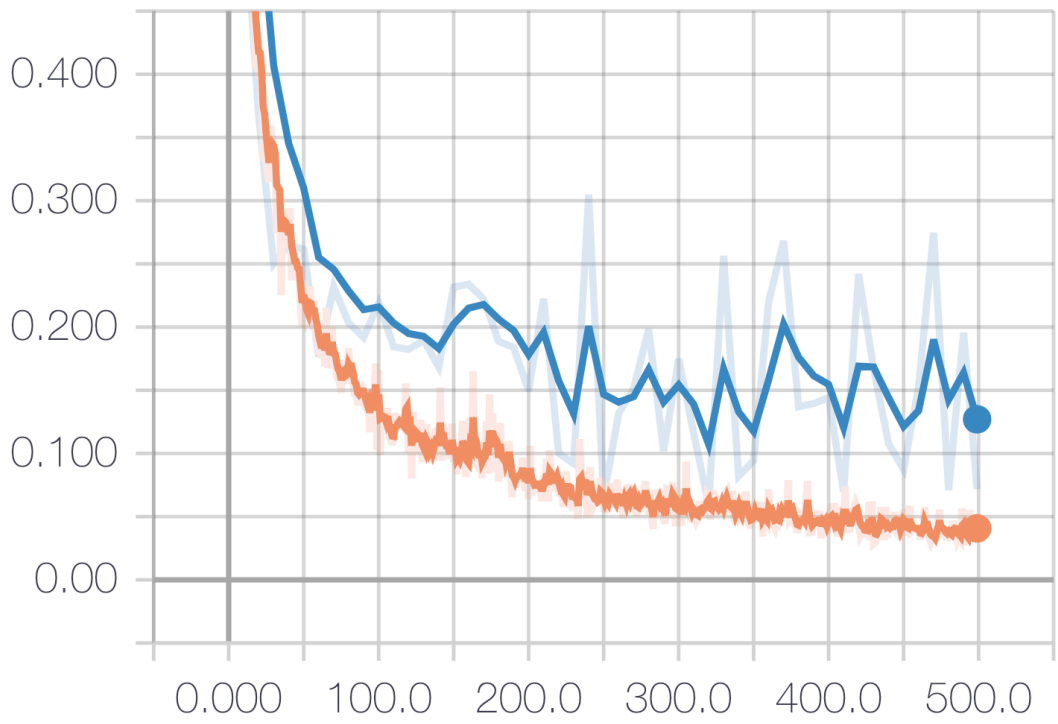

Рисунок 2.12 – Візуалізація значень крос-ентропії, створена за допомогою TensorBoard

Функція втрат – це функція, значення якої використовується для оптимізації значень параметрів в моделі, що розробляється (навчається) [13]. У випадку розробки класифікатора зображень найчастіше використовується крос-ентропія, яка обчислює різницю між очікуваним результатом на виході та реальним отриманим значенням під час тренування і представлена формулою:

$$H(p,q) = -\sum_{i} y_i \log p_i \quad ,$$

де y_i – це очікувана ймовірність приналежності певного зображення до однієї з категорій, а p_i – реально отримана. Крос-ентропія використовується, коли активацію вузла можна представити як вектор ймовірностей того, що зображення відноситься до кожної з категорій, тобто коли виходом є розподіл імовірностей. Таким чином, вона застосовується як функція втрат в нейронних мережах, які мають активацію sigmoid або подібну їй у вихідному

шарі.

Для того, щоб зрозуміти, як саме працює функція втрат, уявимо конволюційну нейронну мережу, що навчена класифікувати зображення за наступними категоріями: собака, кіт, заєць, лиса та вовк. Кожному зображенню ставиться у відповідність мітка (категорія) за допомогою ініціалізації вектору, довжина якого дорівнює кількості класів, а значення можуть бути 1 або 0. Тобто, виходячи з вищесказаного, маємо наступні вектори P_i з мітками для кожного з зображень, які потрапляють на вхід мережі:

- P_1 = [1, 0, 0, 0, 0];
- P_2 = [0, 1, 0, 0, 0];
- P_3 = [0, 0, 1, 0, 0];
- P_4 = [0, 0, 0, 1, 0];
- P_5 = [0, 0, 0, 0, 1].

Нехай наша модель недостатньо натренована і має на виході для тестового зображення собаки вектор ймовірностей [0.4, 0.3, 0.05, 0.05, 0.2], що позначається як Q_1. Такий результат не містить очевидної переваги одного класу над іншим, тому на нього не можна спиратись як на достовірний. Але, для того щоб мережа під час навчання ідентифікувала такий результат як незадовільний, вона повинна отримати значення ентропії. Для цього окремого випадку обчислення є наступними:

$$H(P_1, Q_1) = -\sum_i P_1(i) \log Q_1(i)$$
$$= -(1 \cdot \log 0.4 + 0 \cdot \log 0.3 + 2 \cdot 0 \cdot \log 0.05 + 0 \cdot \log 0.2)$$
$$= -\log 0.4 \approx 0.916\,.$$

Для того щоб зрозуміти, що означає отриманий результат, розглянемо іншу ситуацію, коли мережа достатньо натренована і Q_1 дорівнює $[0.98, 0, 0.01, 0, 0.01]$. Тоді крос-ентропія буде дорівнювати:

$$H(P_1, Q_1) = -\sum_i P_1(i) \log Q_1(i) = -(1 \cdot \log 0.98 + 2 \cdot 0 \cdot \log 0 + 2 \cdot 0 \cdot \log 0.01) = -\log 0.98 \approx 0.02 .$$

Порівнюючи ці два отримані значення, можна помітити, що значення крос-ентропії знижується із зростанням точності прогнозування нейронної мережі і в нормальному випадку має прямувати до нуля під час тренування моделі, як можна побачити на рисунку 2.12.

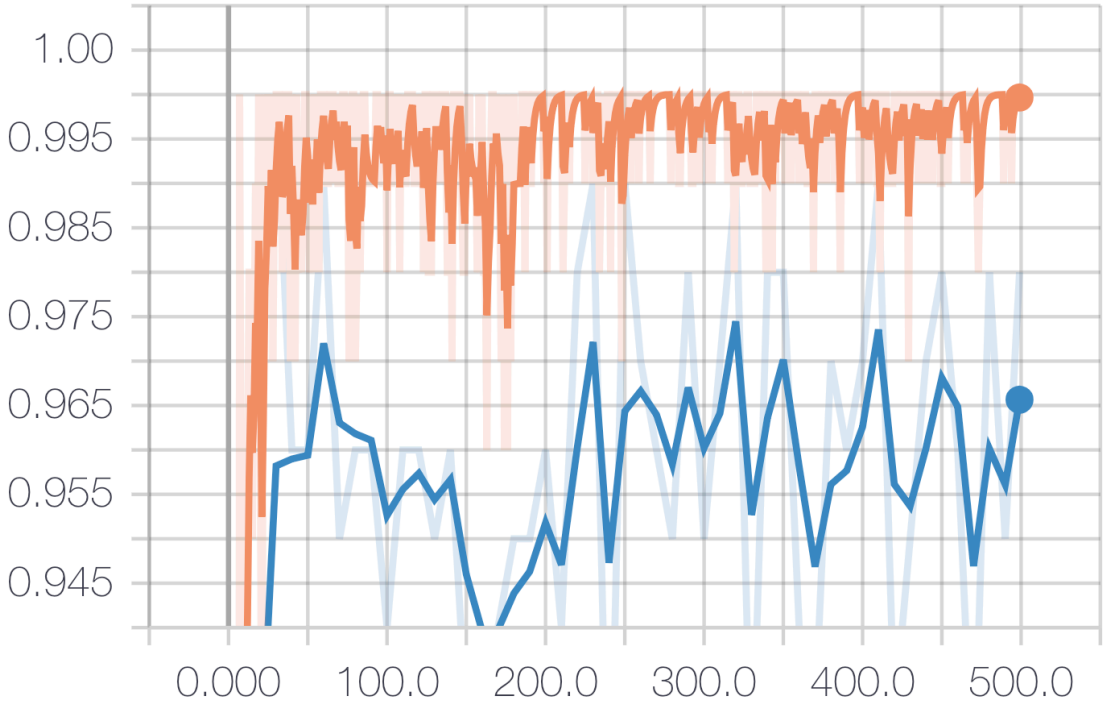

Рисунок 2.13 – Візуалізація точності роботи мережі, створена за допомогою TensorBoard

Графіки на рисунках 2.12 та 2.13 є взаємопов'язаними, що не складно помітити. На рисунку 2.12 верхня лінія графіку відображає значення крос-ентропії на перевірочному наборі даних для навчання, а нижня – на тренувальному. На рисунку 2.13 ситуація з лініями графіку склалась навпаки: верхня лінія відображає точність роботи моделі на тренувальному наборі, а нижня – на перевірочному. Помітно, що при дещо більших значеннях функції втрат точність мережі дещо менша і навпаки.

Ще однією корисною візуалізацією в TensorBoard є гістограма будь-якої змінної моделі (рис. 2.14), яка додається до звіту (інформаційної панелі) за допомогою функції *histogram* з бібліотеки TensorFlow.

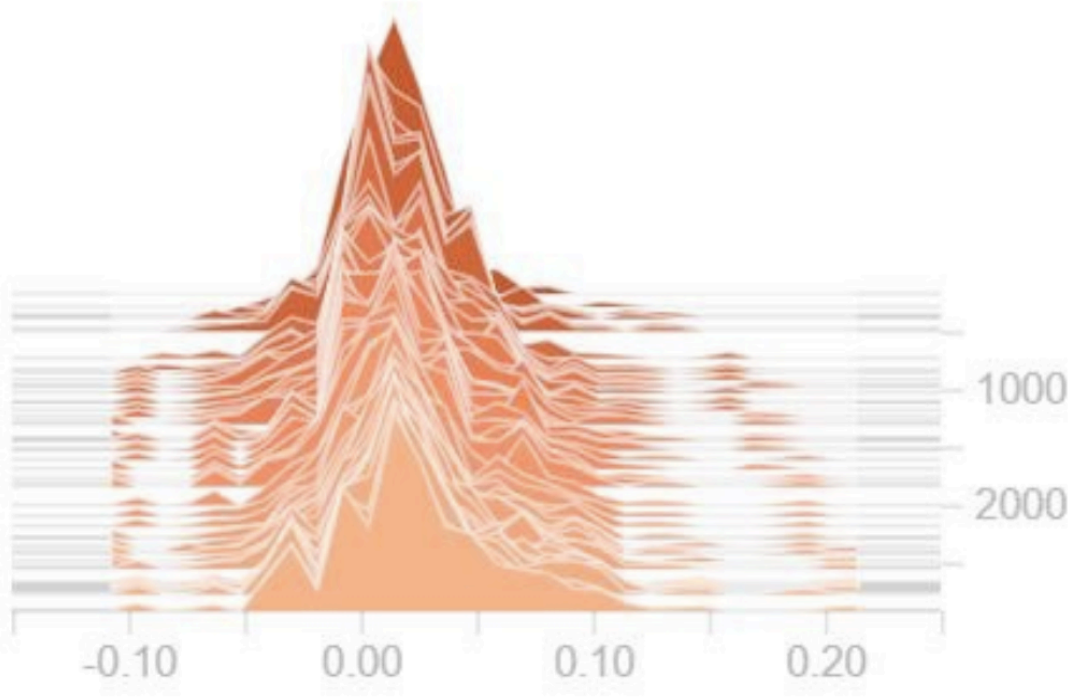

Рисунок 2.14 – Гістограма значень зсуву, створена за допомогою TensorBoard

Гістограма, наведена вище, показує як значення зсуву змінювались під час навчання. Зсув – це константа, яка додається до суми ваг в окремому нейроні. За допомогою цієї константи зсувається функція активації у вузлі під час навчання, що дозволяє нейронній мережі пристосуватись до даних, на яких вона навчається. Кожен фрагмент гістограми є знімком, зробленим в певний момент часу. Фрагмент на передньому плані говорить про те, що після закінчення тренування, зсуви в шарі були розподілені між значеннями -0.05 та 0.1. Фрагменти, розташовані на задньому плані, є більш темними. Вони дозволяють дізнатись, які значення зсуву були раніше у часі, тобто до завершення процесу навчання.

Гістограми дозволяють побачити чи були значення окремих параметрів

мережі змінені належним чином. Для прикладу розглянемо ще одну гістограму, створену для відстеження значень ваг в прихованих шарах (рис. 2.15).

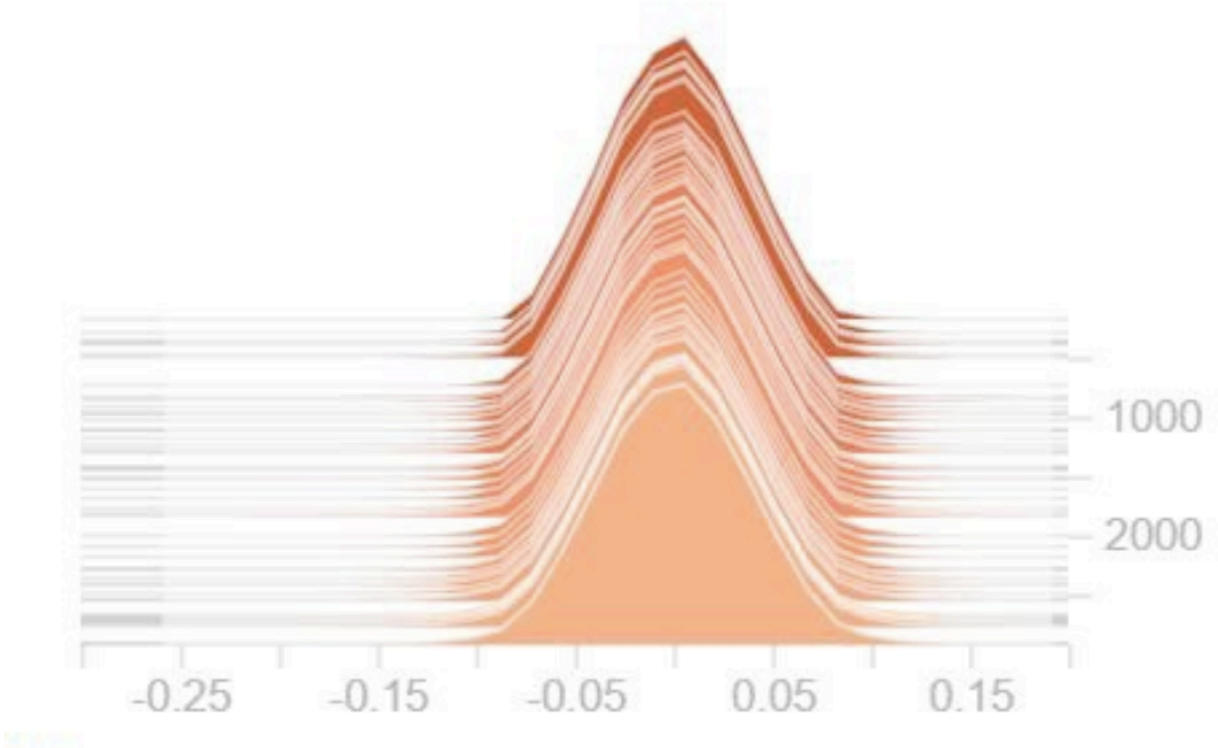

Рисунок 2.15 – Гістограма значень ваг, створена за допомогою TensorBoard

На рисунку вище можна побачити, що значення ваг майже ніяк не змінились в процесі навчання нейронної мережі, що говорить про деякі проблеми, які необхідно вирішувати за допомогою нової конфігурації параметрів моделі.

Отже, можна підвести підсумки проведеного аналізу існуючих підходів та засобів для вирішення задачі класифікації зображень. Порівнявши три існуючи нейронні мережі AlexNet, ResNet та Inception v3, вирішено використовувати останню модель, оскільки на одних й тих самим тестових даних вона в більшості випадків показує кращі результати (найнижча кількість помилкових прогнозувань), а також має меншу кількість параметрів, що дозволяє їй швидше навчатись. Для вирішення задачі саме багатокластерної класифікації запропоновано замінити функцію softmax на

виході моделі Inception v3 на сигмоїдну функцію активації, яка дозволяє класифікувати зображення за багатьма мітками (категоріями), зберігаючи співвідношення між значеннями вектору при перетворенні їх на ймовірності приналежності вхідного образу до кожного із зацікавлених класів. Також, для запобігання ефекту перенавчання мережі вирішено використовувати шар виключення, але тільки в одному місці – після повністю з'єднаного шару моделі, що не призводить до втрати важливих ознак при тренуванні мережі. Для налаштування та аналізу процесу навчання нейронної мережі застосовуються засоби бібліотеки TensorFlow, які дозволяють детально відстежити різноманітні показники.

Висновки до розділу 2

1. Проведено аналіз та порівняння існуючих згорткових нейронних мереж для вирішення задачі класифікації зображень та показано, що їх можна ефективно використовувати для будь-якої конкретної задачі з унікальною сукупністю категорій.
2. На основі проведеного дослідження запропоновано створення нейронної мережі на базі методу перенавчання існуючої моделі Inception v3, що включає в себе:
    а) заміну функції softmax на виході моделі на сигмоїдну функцію активації, яка дозволить класифікувати зображення за багатьма мітками за допомогою зберігання співвідношення між значеннями вектору при перетворенні їх на ймовірності приналежності зображення до певного класу;
    б) додавання шару виключення після повністю з'єднаного для запобігання ефекту перенавчання.

РОЗДІЛ 3. ОПИС РОБОТИ СИСТЕМИ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

В попередньому розділі проведено аналіз та порівняння існуючих моделей для класифікації зображень, з яких обрано Inception v3 для використання при вирішенні поставленої задачі. Також розглянуто різні методи та підходи для боротьби з перенавчанням нейронної мережі, один з яких використовується при створенні класифікатора. Оскільки створювана модель передбачена для класифікації зображень користувачів соціальної мережі з метою їх пошуку за певними критеріями, вона повинна бути здатною до швидкого навчання, тому що неможливо врахувати всі можливі варіанти пошуку й створити універсальну модель. Тому вирішено застосовувати метод передачі навчання для створення власної мережі на основі Inception v3. Це дозволяє скоротити час тренування, а також використовувати невеликий об’єм даних для навчання.

В різних задачах класифікації кількість необхідних зображень, на яких може тренуватись мережа, вимірюється тисячами, а то й десятками тисяч. Звичайно, такий об’єм даних важко підготувати за невеликий проміжок часу. Таким чином необхідно знайти якийсь мінімум потрібних даних, який дозволить навчити мережу ідентифікувати на зображенні те, що вимагається системою пошуку, та ігнорувати зображення, які не відносяться до жодної з категорій. Тому вирішено обрати дещо більшу кількість зображень для кожного класу, ніж та, що рекомендована та вказана в документації методу перенавчання бібліотеки TensorFlow, яка дорівнює 1000. Вирішивши, скільки зображень потрібно підготувати, можна приступати до створення класифікатора.

3.1 Опис використаних категорій класифікації

Для прикладу були обрані три класи (категорії) зображень, серед яких “подорож”, “музика” та “зимовий активний відпочинок”. Тобто створювана модель повинна вміти класифікувати зображення за цими трьома класами, що означає виявлення на фото різноманітних деталей, пов’язаних із ними, та відповідно до цього віднесення вхідного образу до тієї чи іншої категорії. Наприклад, до класу “музика” можна віднести фотографію, на якій зображено якийсь музичний інструмент або будь-яка апаратура, призначена для створення та обробки звукових сигналів тощо (рис. 3.1).

Рисунок 3.1 – Зображення музичного інструмента, яке входить до сукупності даних для навчання нейронної мережі

Якщо деякий відсоток знімків, зроблених користувачем соціальної мережі, містить подібні об’єкти, то можна сказати, що він полюбляє або слухати, або створювати музичні композиції. Таким чином, для кожної з трьох категорій потрібно визначити, які саме деталі планується ідентифікувати на них, адже їх кількість може бути дуже великою. Тому проведено деяке спостереження, використовуючи соціальну мережу Instagram, та вирішено, що для тестування роботи нейронної мережі класи зображень будуть визначені наступним чином. Наприклад, люди, які люблять

або вимушені подорожувати, дуже часто викладають в соціальну мережу фотографії з аеропорту, літаку, поїзду або іншого виду транспорту. На знімках можна побачити сумки в руках або поруч, рюкзак на спині тощо. Тому до категорії “подорож” відноситься зображення, яке містить різні об’єкти, пов’язані з подорожуванням або відпочинком, такі як валіза, сумка різного типу та розміру, фотоапарат, квитки для проїзду на будь-якому виді транспорту, а також сам транспортний засіб. Клас “музика” визначається такими об’єктами на зображенні, як будь-який музичний інструмент (гітара, барабани, клавішні тощо), людина у навушниках або з мікрофоном в руках, музичний гурт на сцені та інші. Категорію “зимовий активний відпочинок” вирішено дещо обмежити і відносити до неї зображення, пов’язані з такими видами діяльності як катання на сноуборді та лижах, тому фотографіями для навчання нейронної мережі є ті, що містять сноуборд, лижи, захисні маски та інші елементи спецодягу.

Підготувавши необхідні дані та створивши відповідну структуру, можна починати навчання моделі.

3.2 Конфігурація згорткової нейронної мережі

Як зазначалось в другому розділі, при створенні нейронної мережі дуже важливим є налаштування її параметрів, від яких залежить, наскільки точні результати буде видавати натренована модель. Одним з найважливіших параметрів, що підбирається емпіричним шляхом, є коефіцієнт швидкості навчання (з англ. “learning rate”), який дозволяє керувати величиною корекції ваг на кожній ітерації тренування. Його значення може бути від 0 до 1 і рекомендованим значенням, як вказано в документації до бібліотеки TensorFlow в розділі про метод передачі навчання, є 0.01. Оскільки для запобігання можливої проблеми перенавчання мережі вирішено використовувати метод виключення, що також впливає на процес

тренування, пропонується розглянути та проаналізувати різні комбінації конфігурацій нейронної мережі.

Перша конфігурація мережі являє собою використання коефіцієнту швидкості навчання, що дорівнює 0.01, без шару виключення. Для того, щоб мати можливість проаналізувати процес навчання, використовується комплект візуалізації TensorBoard, за допомогою якого будуються графіки точності роботи мережі та крос-ентропії (функції втрат). Таким чином, після виконання тренування мережі із конфігурацією, описаною вище, отримано дві візуалізації процесу навчання (рис. 3.2 та 3.3).

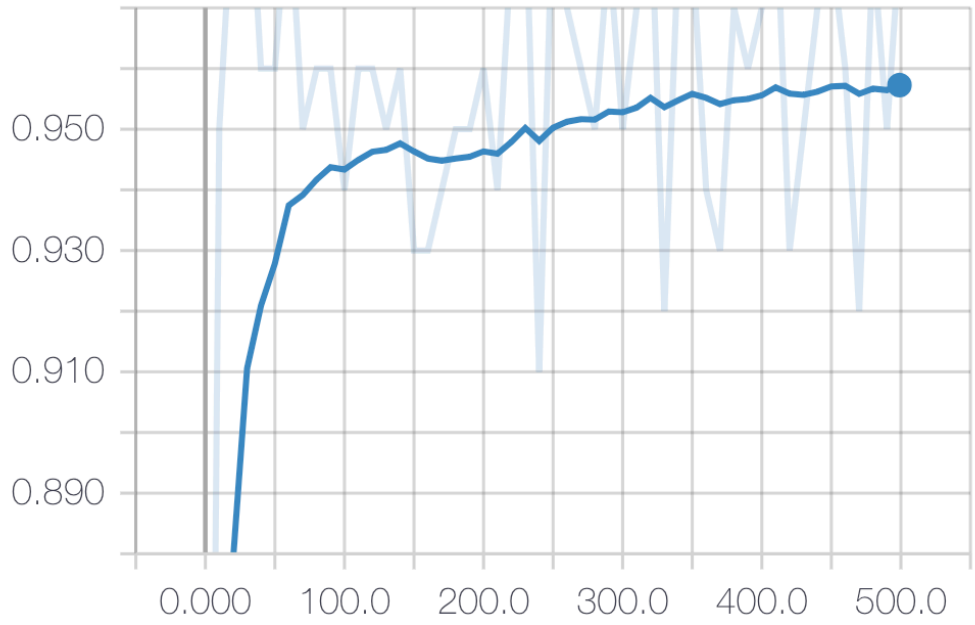

Рисунок 3.2 – Графік точності роботи мережі з коефіцієнтом швидкості навчання 0.01

Вищенаведений графік показує точність роботи нейронної мережі на кожному кроці навчання, тобто ймовірність правильної класифікації, що отримана в результаті обробки зображення, взятого з перевірочного набору даних. Слід нагадати, що до перевірочного набору, який складає приблизно 10% від загального об’єму підготовлених даних, потрапляють зображення, яких мережа ніколи ще не бачила. В результаті тренування мережі отримано точність дещо вище 0.95, що є дуже високим показником. Але слід ще

звернути увагу на значення крос-ентропії.

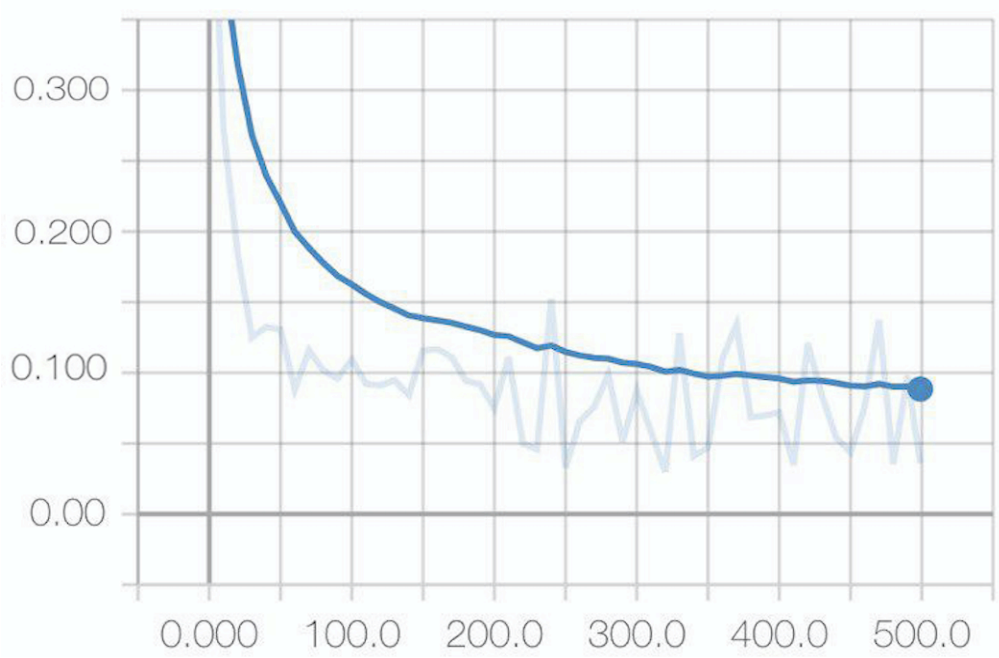

Рисунок 3.2 – Графік спадання значень крос-ентропії в мережі з коефіцієнтом швидкості навчання 0.01

Як видно на графіку в кінці навчання значення крос-ентропії дорівнює близько 0.1. Раніше визначено, що це значення має прямувати до 0 в достатньо натренованих моделях, адже чим воно менше, тим менше помилкових ситуацій отримується на виході мережі.

Результати показали, що нейронна мережа недостатньо навчена, що можна спробувати вирішити зменшенням коефіцієнту швидкості навчання. Тому наступним етапом є створення моделі з меншим значенням цього параметру без додавання шару виключення. Графіки, отримані в результаті цієї конфігурації представлені на рисунках 3.4 та 3.5.

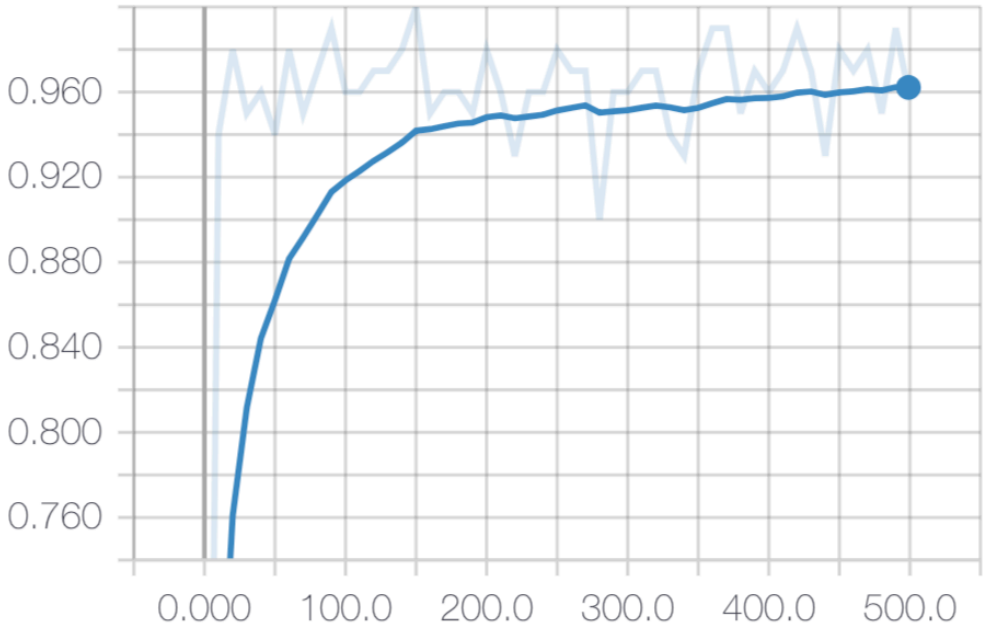

Рисунок 3.4 – Графік точності роботи мережі з коефіцієнтом швидкості навчання 0.005

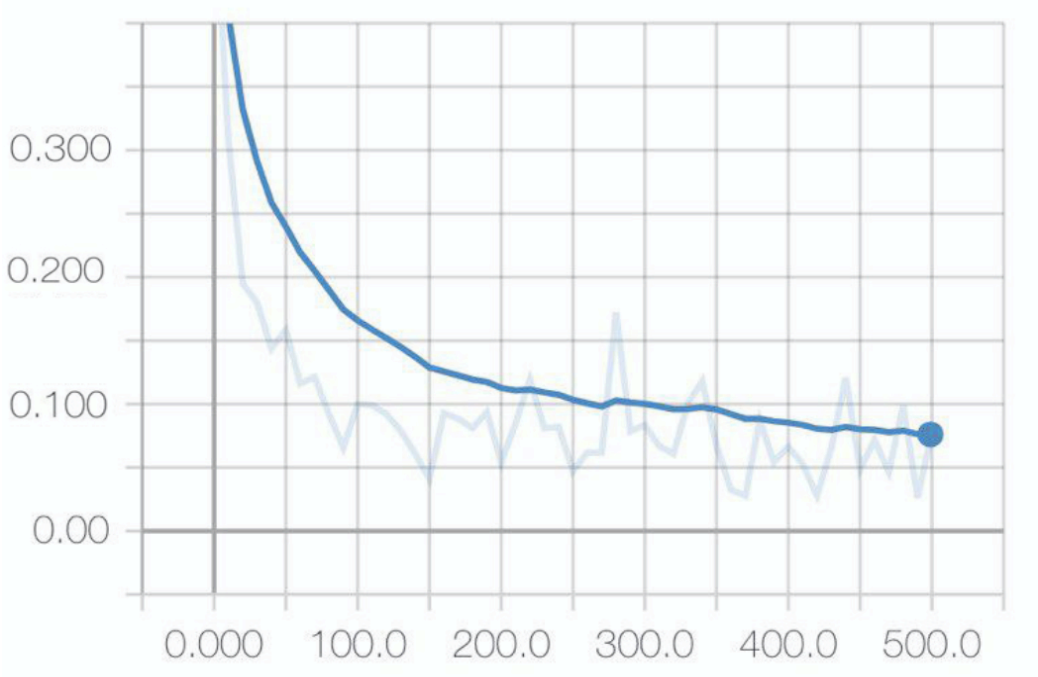

Рисунок 3.5 – Графік спадання значень крос-ентропії в мережі з коефіцієнтом швидкості навчання 0.005

Експериментальним шляхом підібрано значення параметру *learning_rate*, що дорівнює 0.005, при зменшенні якого спостерігається ефект

перенавчання. Графіки, отримані в результаті навчання моделі, показують, що модель має дещо більшу точність класифікації зображень (приблизно 0.96) та менше значення крос-ентропії. Але можна покращити ці показники за допомогою застосування методу виключення, що видно на наступному графіку (рис. 3.6).

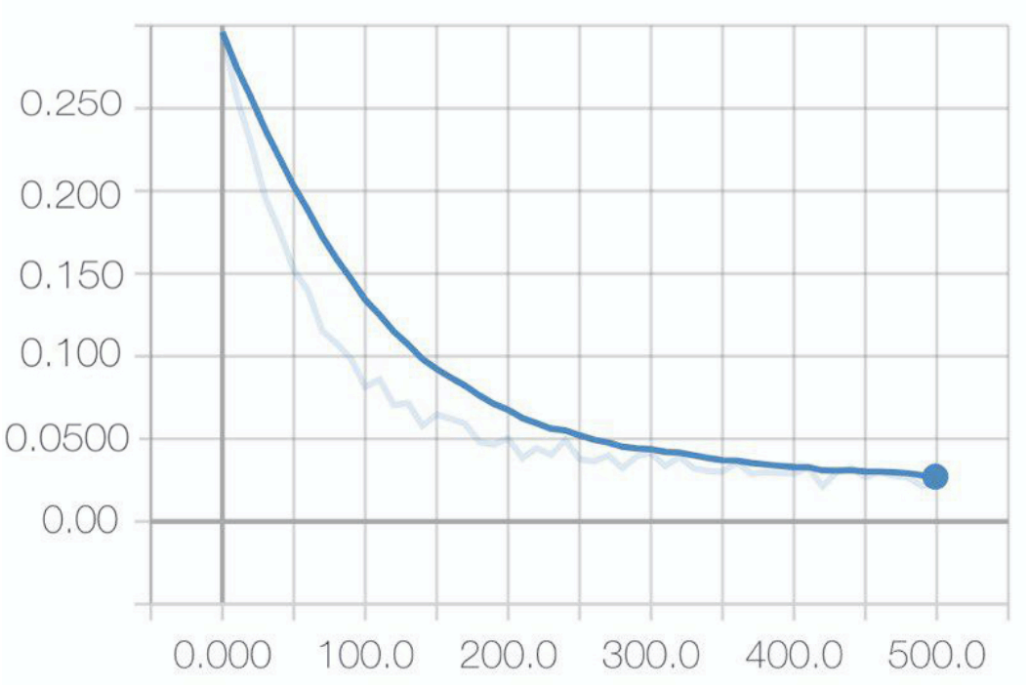

Рисунок 3.6 – Графік спадання значень крос-ентропії в мережі з шаром виключення та з коефіцієнтом швидкості навчання 0.005

Останній варіант конфігурації мережі з коефіцієнтом швидкості навчання 0.005, який полягає у використанні шару виключення після повністю з'єднаного, виявився найвдалішим. Значення крос-ентропії в кінці тренування має порядок 10^{-2} і дорівнює приблизно 0.02. Точність класифікації залишилась такою ж самою, як в попередньому випадку (рис. 3.7), і дорівнює приблизно 0.96.

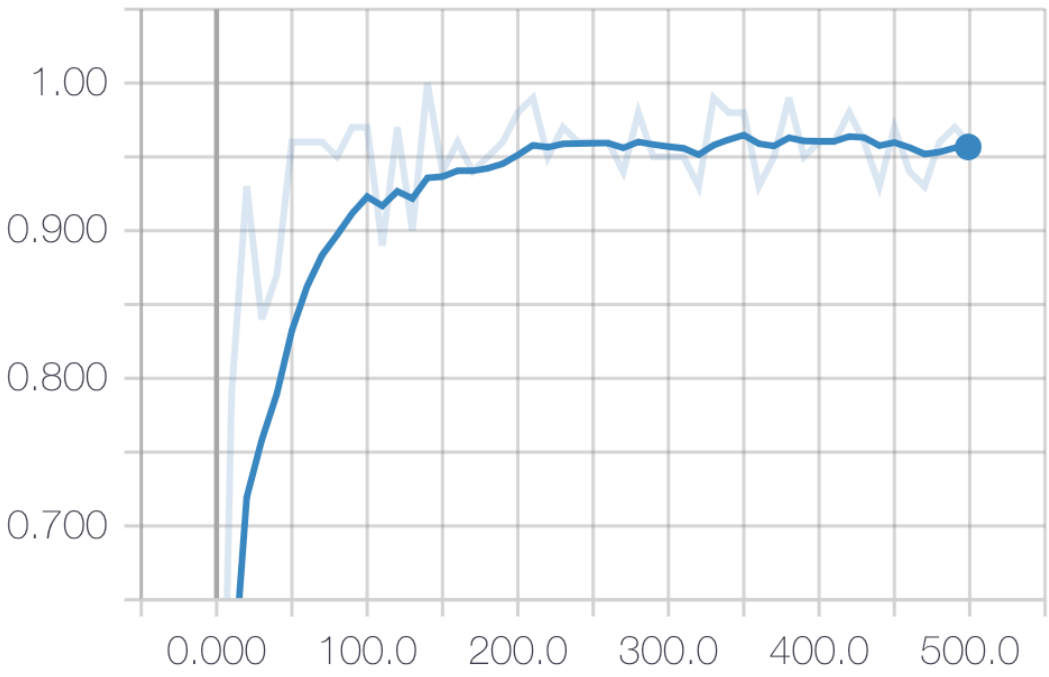

Рисунок 3.7 – Графік точності роботи мережі з шаром виключення та з коефіцієнтом швидкості навчання 0.005

Модель останньої конфігурації має найкращі результати навчання. З графіку видно, що одразу на початкових ітераціях не було отримано точності близької до 1, що також говорить про відсутність ефекту перенавчання. Тому для класифікації зображень далі використовується модель саме з такими параметрами.

Візуалізації, створені за допомогою TensorBoard, показали, що кінцева модель достатньо натренована та може видавати результати високої точності. Для того, щоб перевірити роботу мережі, пропонується протестувати її на декількох зображеннях, взятих із соціальної мережі Instagram.

3.3 Тестування роботи створеного класифікатора

Спочатку використаємо зображення, на яких знаходяться об’єкти тільки одного класу, для того, щоб впевнитись, що модель не видає високих результатів щодо інших двох класів. Першим тестовим зображенням є

фотографія користувача, зроблена в аеропорту. На другій фотографії зображені три чоловіка з гітарою, а на третій – чоловік на сноуборді. Всі три фотографії наведені на рисунку 3.8.

Рисунок 3.8 – Тестові зображення для перевірки роботи створеної нейронної мережі

Для першого тестового випадку мережа видала наступні результати: клас “подорож” – ймовірність 0.92, клас “музика” – 0.17, клас “зимовий активний відпочинок” – 0.09. Для другого зображення ймовірності розподілились наступним чином: 0.2, 0.95 та 0.13. Третю фотографію модель віднесла до класу “подорож” з ймовірністю 0.34, до класу “музика” з ймовірністю 0.08 і до класу “зимовий активний відпочинок” з ймовірністю 0.93.

Для того, щоб впевнитись, що модель здатна до багатокластерної класифікації, обрано зображення, яке можна віднести одразу до двох категорій (рис. 3.9).

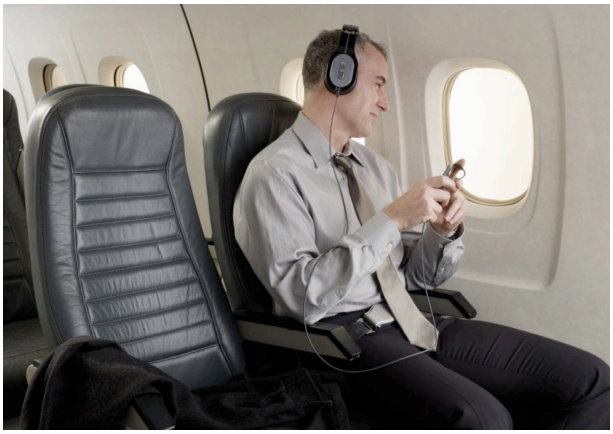

Рисунок 3.9 – Тестове зображення з двома мітками (“подорож” і “музика”)

Для наведеного зображення отримані наступні ймовірності приналежності до кожного з класів:

- клас “подорож” – 0.82;
- клас “музика” – 0.85;
- клас “зимовий активний відпочинок” – 0.05.

Отже, проаналізувавши отримані результати роботи створеної нейронної мережі, можна встановити поріг ймовірності, за значенням якого система вирішує, чи достовірним є кожне зі значень на виході. Для всіх тестових випадків, ймовірність приналежності зображення до очікуваного класу є більшою за 0.8, як при класифікації за однією міткою, так і при багатокластерній класифікації. Тому, якщо задати значення порогу, яке дорівнює 0.8, і ігнорувати ймовірності, які менші цього значення, то можна спиратись на результати, отримані на виході моделі, як на достовірні.

Висновки до розділу 3

Розроблено систему багатокластерної класифікації зображень та проаналізовано характеристики процесу навчання нейронної мережі з різними конфігураціями, тобто з різними значеннями параметрів, та отримано наступні результати.

1. Розроблена модель із заданим значенням коефіцієнту швидкості навчання 0.01 має точність близько 0.95 та високе значення крос-ентропії, яке дорівнює 0.1, що говорить про недостатню натренованість.
2. Експериментальним шляхом підібрано значення коефіцієнту швидкості навчання, яке дорінює 0.005, при зменшенні якого в моделі спостерігається ефект перенавчання. Значення крос-ентропії при цьому дещо зменшилось і дорівнює 0.08.
3. Застосовано шар виключення, який дозволив зменшити значення крос-ентропії приблизно в 4 рази – до 0.02, та досягти точності роботи мережі, що становить 0.96. Таким чином створено нейронну мережу, яка має порівняний відсоток помилкових результатів з моделю Inception v3, до якої застосований метод передачі навчання, але на відміну від неї дозволяє класифікувати зображення за декількома критеріями.

ВИСНОВКИ

1. Проведено аналітичний огляд методів класифікації зображень та показано, що існуючі рішення не є універсальним розв’язком, тобто кожна задача з унікальними категоріями класифікації потребує власного рішення або реалізації відомих алгоритмів.
2. Запропоновано підхід для вирішення поставленої задачі, який полягає у використанні методу передачі навчання на основі існуючої згорткової нейронної мережі, тобто заміни лише одного з багатьох її шарів, що потребує менших обчислювальних ресурсів та часу на реалізацію, порівняно зі створенням нового алгоритму.
3. Проаналізовано та порівняно три різні згорткові нейронні мережі, серед яких обрана модель Inception v3 на основі показників точності роботи, кількості параметрів, а також архітектурного рішення, яка дозволяє отримати точність класифікації, тобто ймовірність правильного віднесення зображення до того чи іншого класу, близько 0.95.
4. Вирішено використовувати іншу функцію активації на виході моделі – сигмоїдну, яка дозволяє класифікувати зображення за багатьма мітками за допомогою зберігання співвідношення між значеннями вихідного вектору після перетворення їх у ймовірності, що не робить їх такими, щоб сума цих ймовріностей дорівнювала 1, як це відбувається у випадку використання функції softmax.
5. Порівняно роботу нейронної мережі з різними конфігураціями, значеннями параметрів, що задаються, та показано, що модель з коефіцієнтом швидкості навчання 0.005 та використанням шару виключення має найкращі показники точності та помилкових результатів. Створена модель здатна класифікувати зображення за багатьма мітками.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. В. Штеньович. Вступ до машинного навчання / В. Штеньович, А. Білоус // Режим доступу: https://dou.ua/lenta/articles/introduction-machine-learning-1
2. M. Seetha. Comparison of Advanced Techniques of Image Classification / M. Seetha, V. Muralikrishna // Режим доступу: https://www.researchgate.net/publication/237609215_Comparison_of_Advanced_Techniques_of_Image_Classification
3. N. Bartyzal. Artificial neural networks // Режим доступу: http://shodhganga.inflibnet.ac.in/bitstream/10603/48/6/chaper%204_c%20b%20bangal
4. A. Moawad. Neural networks and back-propagation algorithm // Режим доступу: https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e
5. S. Hijazi. Convolutional Neural Networks for Image Recognition / S. Hijazi, R. Kumar // Режим доступу: https://ip.cadence.com/uploads/901/cnn_wp-pdf
6. N. Donges. Transfer Learning // Режим доступу: https://towardsdatascience.com/transfer-learning-946518f95666
7. J. Deng. ImageNet: A large-scale hierarchical image database / J. Deng, W. Dong, R. Socher // Режим доступу: https://ieeexplore.ieee.org/document/5206848
8. J. Jordan. Common architectures in convolutional neural networks // Режим доступу: https://www.jeremyjordan.me/convnet-architectures
9. E. Culurciello. Neural Network Architectures. Inception model // Режим доступу: https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

10. V. Fung. An overview of ResNet and its variants // Режим доступу: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035
11. S. Sharma. Activation functions: neural networks // Режим доступу: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
12. R. Ruizendaal. Deep learning: hadling overfitting // Режим доступу: https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d
13. P. Dahal. Classification and Loss Evaluation – Softmax and Cross Entropy Loss // Режим доступу: https://deepnotes.io/softmax-crossentropy