МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна магістерська робота

на тему:

«Інформаційна технологія оптичного розпізнавання тексту з зображення для мобільного додатку»

Завідувач кафедрою **Довбиш А.С.**

Керівник роботи **Москаленко В. В.**

Студент групи ІН м.-02 **Потапенко Д. О.**

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *122 «Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Потапенко Дарині Олексіївні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна система розпізнавання зображень з підвищеним рівнем робастності до шуму.*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні дані до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми та постановка задачі 2) Вибір методу 3) Інформаційна та програмна реалізація системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання

_____

Керівник_____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми та постановка задачі*		
2.	*Вибір методу*		
3.	*Інформаційна та програмна реалізація системи*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 45 стор., 18 рис., 1 табл., 1 додаток, 11 джерел.

Об’єктом дослідження є методи і алгоритми реалізації методу contrastive center loss у задачах розпізнавання зображень.

Мета роботи — застосування методу contrastive center loss у задачах розпізнавання зображень, а саме — реалізація нейромережевої структури для розпізнавання об'єктів на зображеннях з підвищеним рівнем робастності до шуму.

Методами дослідження є методи і алгоритми реалізації методу contrastive center loss у задачах розпізнавання зображень.

У якості **результатів** реалізовано нейромережеву структуру для розпізнавання об'єктів на зображеннях з підвищеним рівнем робастності до шуму. Даний алгоритм реалізовано у формі програмного забезпечення, створеного за допомогою згорткових нейронних мереж (ЗНМ, convolutional neural network).

РОЗПІЗНАВАННЯ, ІНІЦІАЛІЗАЦІЯ НЕЙРОМЕРЕЖІ, АРХІТЕКТУРА НЕЙРОННИХ МЕРЕЖ, РОБАСТНІСТЬ ДО ШУМУ

ЗМІСТ

ВСТУП

“Роками вченими робилися спроби відтворити здатність біологічної нервової системи навчатися та виправляти помилки, що призвело до створення штучних нейронних мереж”. [1] Штучні нейронні мережі – математичні моделі, а також їх апаратні та програмні втілення, що будуються за принципом організації та функціонування біологічних НМ. На сьогоднішній день нейромережеві структури успішно застосовуються для вирішення різноманітних завдань, таких як: кластеризація, адаптивне управління, розпізнавання мови, машинний зір та ряд інших не менш важливих задач.

Робота із зображеннями – важлива сфера застосування технологій машинного навчання. Глобально усі зображення становлять бібліотеку неструктурованих даних. Задіявши НМ системи, машинне навчання та штучний інтелект, ці дані групуються та використовуються для виконання різних задач: в побутових, соціальних, професійних та державних потребах.

Проблематика обробки та розпізнавання зображень відноситься до важко формалізованих завдань, і є одним з найбільш актуальних на сьогоднішній день. Для вирішення цієї задачі активно розробляються різноманітні архітектури нейронних мереж, що дають значно точніші результати порівняно з іншими алгоритмами.

За розпізнаванням образів стоїть майбутнє розвитку штучного інтелекту. Тому задача вирішення проблеми чутливості нейронних мереж до шумів на вхідних даних з часом стає все більш актуальною.

Мета роботи — застосування методу contrastive center loss у задачах розпізнавання зображень, а саме — реалізація нейромережевої структури для

розпізнавання об'єктів на зображеннях з підвищеним рівнем робастності до шуму.

Для досягнення зазначеної мети **вирішено наступні завдання:**

1. Розглянути та проаналізувати існуючі підходи до вирішення задач підвищення узагальнюючих можливостей нейронних мереж.
2. Провести порівняльну характеристику існуючих підходів до підвищення узагальнюючих властивостей нейронних мереж.
3. Обрати та детально описати використану в роботі інформаційну модель, методи та алгоритми її навчання.
4. Створити модель результуючої нейронної мережі.
5. Проаналізувати роботу та описати результати дослідження.

Об’єктом дослідження є метод contrastive center loss у задачах розпізнавання зображень.

Предметом дослідження є реалізація та апробація результуючої моделі нейронної мережі.

Методом вирішення поставленої задачі є застосування функції розпізнавання об'єктів з використанням штучних нейронних мереж в умовах зашумленості вхідного зображення.

Результати дослідження — реалізована нейромережева структура для розпізнавання об'єктів на зображеннях з підвищеним рівнем робастності до шуму.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ РОЗПІЗНАВАННЯ ОБРАЗІВ

1.1 Сучасний стан та тенденція розвитку інтелектуальних алгоритмів розпізнавання зображень

З розвитком технологій все більш актуальним постає питання вдосконалення та спрощення роботи комп'ютерної галузі. Людство стрімко рухається в бік нової інформаційної революції, яку можна порівняти за масштабами лише з розвитком Інтернет технологій. Наразі штучний інтелект стрімко розвивається і вивчається. Тут вирішуються конкретні питання, пов'язані зі шляхом розвитку наукової думки в таких областях як: обчислювальна техніка, робототехніка та їх вплив на життя людини. Саме тому побудова систем машинного навчання є, на сьогоднішній день, одним з найсучасніших та найперспективніших областей в розвитку сучасної кібернетики. Зростання попиту та прискорення темпів розвитку “розумних” технологій з часом привертають все більше уваги що й відводить цим галузям особливе місце у науковому світі.

У машинному навчанні аналіз візуальних об'єктів є життєво важливою складовою завдань: розпізнавання, пошуку та класифікації. На сьогоднішній день існує ряд напрямків науки та техніки, які значною мірою орієнтовані на розвиток та вдосконалення систем, що аналізують інформацію, подану у вигляді зображень. Проблематика обробки та розпізнавання зображень відноситься до важко формалізованих завдань, і є одним з найбільш актуальних питань на сьогоднішній день. На даний момент існує безліч методів та алгоритмів вирішення задач розпізнавання об'єктів проте всі вони

поступаються точністю, простотою та швидкістю результатів штучним нейронним мережам.

Робота із зображеннями – важлива сфера застосування технологій машинного навчання. Глобально усі зображення становлять бібліотеку неструктурованих даних. Задіявши НМ системи, машинне навчання та штучний інтелект, ці дані групуються та використовуються для виконання різних задач: в побутових, соціальних, професійних та державних потребах. В даний час для пристроїв цифрової обробки сигналів характерне постійне зростання обсягу оброблюваної інформації, підвищення вимог до якості обробки, робота зі складними “зашумленими” зображеннями. Це стимулює появу нових методів і алгоритмів з обробки та аналізу зображень.

Значний внесок у розробку теорії та алгоритмів ЦГЗ належить як вітчизняним вченим В.А. Котельникову, Я.З. Ципкіну, Л.М. Гольденберг, Л.П. Ярославський, Ю.В. Гуляєва, Ю.Б. Зубарєву, А.А. Ланне, Ю.А. Брюханову, В.В. Витязеву, В.П. Дворкович, А.Т. Мінгазіну, так і зарубіжним Л. Рабінеру, Б. Голду, А. Оппенгейму, Р. Шаферу, В. Каппелліні, Д. Кайзеру, С. Мітре, Р. Гонсалесу, І. Пітасу, А. Венетсанопулусу, Е. Догерті, Я .Астоле, Т.Сарамакі.

Велике значення в системах ЦОЗ відіграють саме нейромережеві структури (НС), сфера застосування яких останнім десятиліттям різко розширилася з переходом від класичних лінійних алгоритмів обробки сигналів та зображень до нелінійних алгоритмів, де нейромережеві засоби займають одне з провідних місць. Це стосується задач відновлення цифрових сигналів та зображень, ідентифікації, сегментації, розпізнавання образів. В останні роки НС стали активно застосовуватися також для побудови систем кодування та декодування сигналів у системах стиснення зображень.

“Теорія НС була створена та розвинена такими вченими, як Ф. Розенблатт, М. Мінський, С. Гроссберг, Т. Кохонен. Сьогодні питаннями теорії та практичного використання НС присвячені роботи російських учених А.І. Галушкіна, А.М. Горбаня, О.М. Балухто, В.І. Горбаченка, В.В. Золотарьова, Н.І. Черв'якова”.[2]

Як основні та найбільш значущі переваги нейромережевої обробки цифрових сигналів та зображень можна виділити наступні:

1. можливість досягнення потенційно вищого відношення «продуктивність/вартість» для нейромережевих обчислювальних засобів у порівнянні з «традиційними» обчислювальними засобами;
2. можливість у рамках єдиного методико-алгоритмічного базису вирішувати різні завдання обробки об'єктів та зображень;
3. можливість ефективного вирішення завдань в умовах неповної апріорної інформації.

Важливо відзначити, що більшість завдань обробки зображень допускають «природний» паралелізм обчислень у реалізації відповідних обчислювальних процедур через специфіку представлення самого цифрового зображення.

Розпізнавання образів належить до однієї з найважливіших задач штучного інтелекту. Вона пов'язана з безліччю різних областей досліджень. Сьогодні задачі розпізнавання — одні з основних завдань, що широко використовуються у комп'ютерному зорі. Розпізнавання образів на зображеннях і вилучення ознак також є важливою частиною інших, складніших методів комп'ютерного зору таких як виявлення об'єктів і сегментація зображення.

В цілому виділяють три види задач систем комп'ютерного зору:

1. Розпізнавання (навчання з учителем) - віднесення об'єктів, що до певних класів за допомогою застосування відомих правил класифікації. Це найбільш типове завдання систем розпізнавання. Перед тим, як система може виконувати цю функцію, передбачається її навчання на безлічі прикладів. навчальній вибірці об'єктів розпізнавання та ін.
2. Класифікація (навчання без вчителя) - розбиття безлічі об'єктів на класи, що не перетинаються, за їх формалізованими описами. Це завдання вирішується у тих випадках, коли від системи не вимагається віднесення вхідних образів до будь-яких певних класів, а необхідна лише здатність розрізняти їх будь-яким способом за певними ознаками.
3. Створення формалізованого опису об'єктів розпізнавання, придатного для використання алгоритмами власного розпізнавання. Як правило, вихідні дані об'єктів, що спостерігаються, представлені у формі, непридатній безпосередньо для розпізнавання. Це можуть бути растрові зображення, звукові файли, статистичні дані (числові набори), відео-записи тощо. Такі алгоритми розпізнавання зазвичай вимагають більш високорівневого уявлення. Це призводить до необхідності зробити одне або більше перетворення вихідних даних.

В якості прикладу можна розглянути процедуру сегментації зображення, тобто, виділення на ньому однотипних областей. Розпізнавання

полягає у знаходженні на зображенні характерного заданим критеріям об'єкту. Ці задачі можуть вирішуватись наступні проблеми:

1. **Ідентифікація**: розпізнається індивідуального екземпляру об'єкта.
2. **Пошук зображень по заданим критеріям**: знаходження всіх зображень у великому наборі даних, що мають певний зміст. Зміст визначається відповідно до обраних критеріїв пошуку.
3. **Оптичне розпізнавання символів:** розпізнавання символів на зображеннях друкованого чи рукописного тексту.

Завдання відновлення зображень вирішується видаленням шуму з зображення (шум датчика, розмитість об'єкта, що рухається і т. д.). Теорія та моделі комп'ютерного зору використовуються для створення систем комп'ютерного зору.

1.2 Існуючі підходи до підвищення узагальнюючих властивостей нейронних мереж

У машинному навчанні, незалежно від того, стикаємося ми з проблемою класифікації або регресії, вибір моделі навчання є надзвичайно важливим, оскільки саме він значною мірою впливатиме на шанс отримання гарного результату. Цей вибір може залежати від багатьох змінних: кількості даних, розмірності простору, гіпотези розподілу. Наразі розроблено велику кількість підходів до навчання, кожен з яких має свої сильні та слабкі сторони, однак до сих пір не існує єдиного алгоритму, здатного вирішувати усі поставлені задачі. У цьому розділі запропоновано розглянути декілька з них.

“Ансамблювання (ансамбль алгоритмів). Ансамбль алгоритмів (методів) — метод, який використовує кілька навчальних алгоритмів з метою отримання кращої ефективності прогнозування”. [3]

“Ансамблеві методи - це парадигма машинного навчання, у яких декілька моделей «слабких учнів» які паралельно навчаються для вирішення однієї проблеми і об'єднуються для отримання найоптимальніших результатів. Основна гіпотеза ансамблювання полягає у тому, що у результаті поєднання слабких моделей ми маємо змогу отримати більш чіткі та точні результати”.[4]

Ансамбль машинного навчання складається з конкретного кінцевого набору альтернативних методів, але, як правило, припускає наявність більш гнучкої структури серед цих альтернатив.

Щоб реалізувати ансамблевий метод, необхідно підібрати декілька моделей «слабких учнів» для агрегування. Найчастіше (у тому числі в добре відомих методах бегінга та бустингу) використовується єдиний базовий алгоритм навчання, а саме: ми маємо дві однорідні системи, які навчаються по-різному. Отримана нами модель ансамблю називається однорідною. Тим не менш, існують інші методи, що використовують різні типи базових алгоритмів навчання типу: різнорідні слабкі учні поєднуються в «різнорідну ансамблеву модель».

Не менш важливою задачею ансамблювання є вибір слабких учнів для агрегування, оскільки вибір моделей для навчання має бути узгоджений з тим, як саме об'єднуються ці моделі. Вибір учнів з низьким зсувом, але високим розкидом, має бути поєднаний за допомогою методу агрегування, котрий має тенденцію зменшувати розкид, тоді як вибір моделі з низьким

розкидом, але з високим зсувом, має поєднуватися з моделями що зменшують зсув.

Таким чином необхідно дізнатися як саме поєднуються ці вибірки. Існує два основних типи мета-алгоритмів, спрямованих на об'єднання слабких учнів. Нижче розглянемо деталі концепції кожного з них.

Бустинг. Перш ніж говорити про бустинг слід згадати терміни «data mining»: сильна та слабка моделі. Сильною називається модель, здатна допустити мінімальну кількість помилок в задачах класифікації, слабка ж навпаки досить часто помилятися - тобто не є точною (втрачає в надійності).

Бустингом (від англ. Boosting - посилення) називається метод, спрямований на перетворення слабких моделей в сильні шляхом побудови ансамблю класифікаторів.

“При бустингу відбувається послідовне навчання класифікаторів. Таким чином, навчальний набір даних на кожному наступному кроці на пряму залежить від точності прогнозування попереднього базового класифікатора”. [4]

Сучасна модифікація алгоритму Boost 1 передбачає використання необмеженої кількості класифікаторів, кожен із яких навчається на одному наборі даних, по черзі застосовуючи їх на різних кроках навчання. Модель методу бустингу зображена на Рисунку 1.1.

Рисунок 1.1 — Модель методу Бустингу

Отже, суть бустингу, як і інших ансамблевих алгоритмів, полягає у тому, щоб із декількох слабких моделей створити одну сильну. Загальна ідея алгоритмів Бустинга – послідовне застосування предикторів, щоб кожна наступна модель мінімізувала помилку попередньої.

“Беггінг – технологія класифікації, де всі елементарні класифікатори навчаються і працюють паралельно (незалежно один від одного). Ідея беггінгу полягає у тому, що класифікатори не виправляють помилки одне одного, а лише компенсують їх при голосуванні”. [4]

Беггінг складається з кількох базових моделей, навчених на різних бутстреп вибірках та побудови ансамблевой моделі, яка «усереднює» результати слабких учнів.

Бутстреп. Цей статистичний метод полягає у генерації вибірок розміру B (так званих бутстреп вибірок) з вихідного датасета розміру N шляхом випадкового вибору елементів з повтореннями на кожному спостереженні B. Модель методу бутстреп можна побачити на Рисунку 1.2.

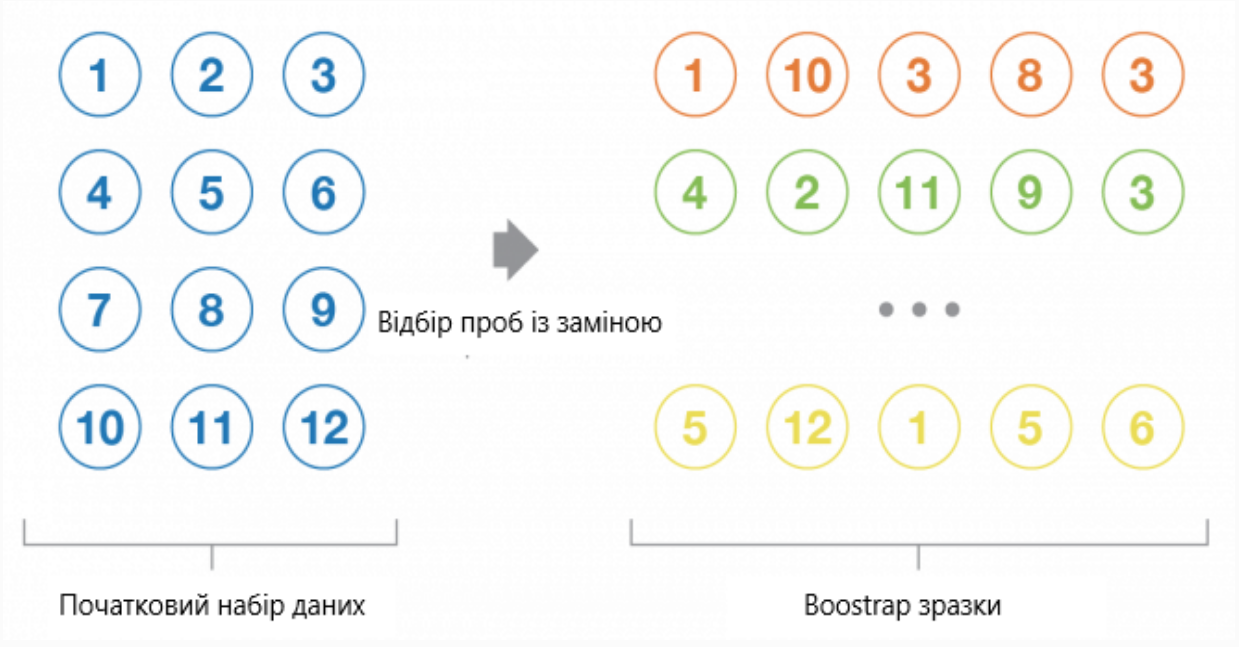

Рисунок 1.2 — Модель методу Бутстреп

Бегінг (bootstrap aggregating) використовує паралельне навчання базових класифікаторів та працює наступним чином:

1. “З множини вихідних даних випадковим чином відбирається декілька підмножин, що містять кількість прикладів, які відповідають кількості вихідної множини. Оскільки відбір здійснюється випадковим чином, набір прикладів завжди відрізнятиметься: деякі приклади потраплять до кількох підмножин, інші не потраплять до жодної.” [3]
2. За підсумками кожної вибірки будується класифікатор.
3. Висновки класифікаторів агрегуються шляхом голосування чи усереднення.

Приклад беггінгу можна спостерігти на Рисунку 1.3.

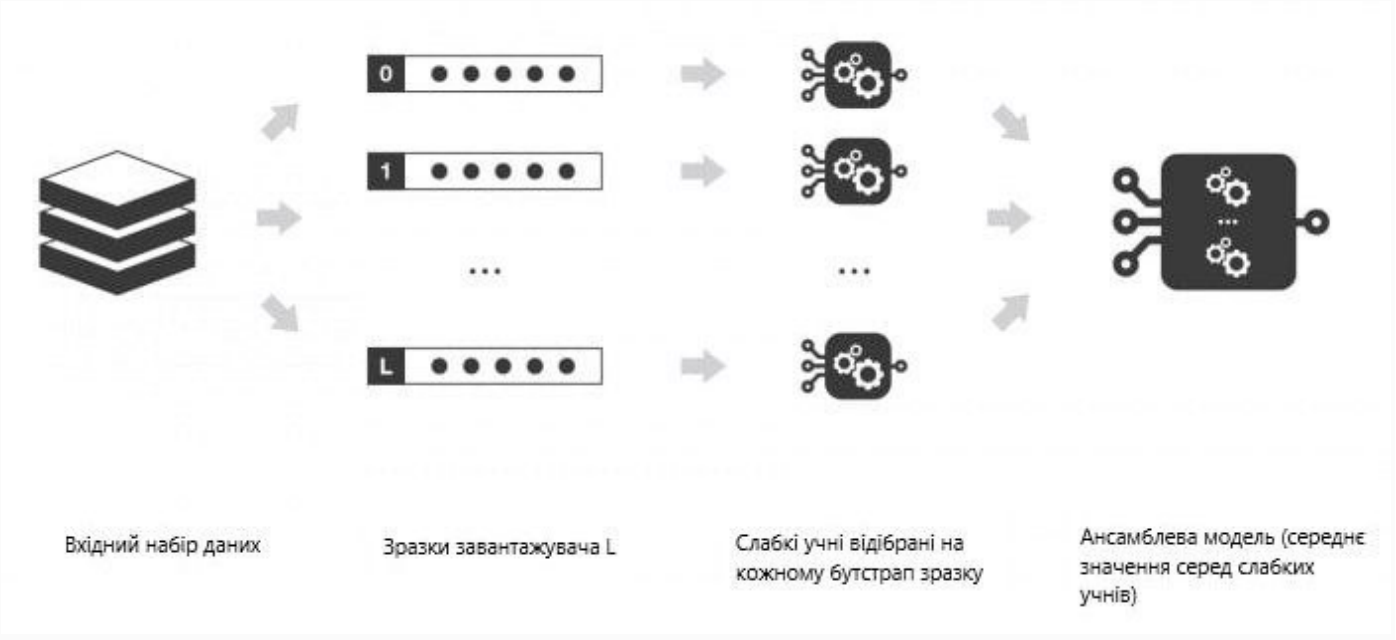

Рисунок 1.3 — Модель технології Беггінгу

Як і при бустингу, очікується, що результат прогнозу агрегованого класифікатора будуть набагато точнішим за результат прогнозу одиночної моделі на тому ж наборі даних.

Регуляризація в машинному навчанні - це методика що базується на додаванні деяких додаткових обмежень до умов задачі та використовується для зменшення помилки шляхом належного підбору функції на заданому навчальному наборі. Позбавляючись шумів, вона допомагає моделі звертати увагу лише на ті ознаки, які дійсно мають значення.

Зачасту фахівці в області машинного навчання використовують регуляризацію для того, аби переконатися, що модель приймає рішення на підставі незалежних даних, що значною мірою впливають на результуючу змінну.

Усі типи регуляризації мають штраф, що позначається грецькою літерою лямбда: λ. Саме він впливає на зменшення кількості шумів у вибірках даних.

У **рідж-регресії** (L2-регресія), штраф - сума квадратів коефіцієнтів. Коефіцієнти лінійної регресії - числа, прикріплені до кожної незалежної змінної, що визначають її вплив на результуючі дані. У рідж-регресії штрафи зменшують коефіцієнти за відповідних незалежних змінних, однак ніколи не обнуляють їх. Це означає, що зашумленість завжди, не значною мірою, впливатиме на результат.

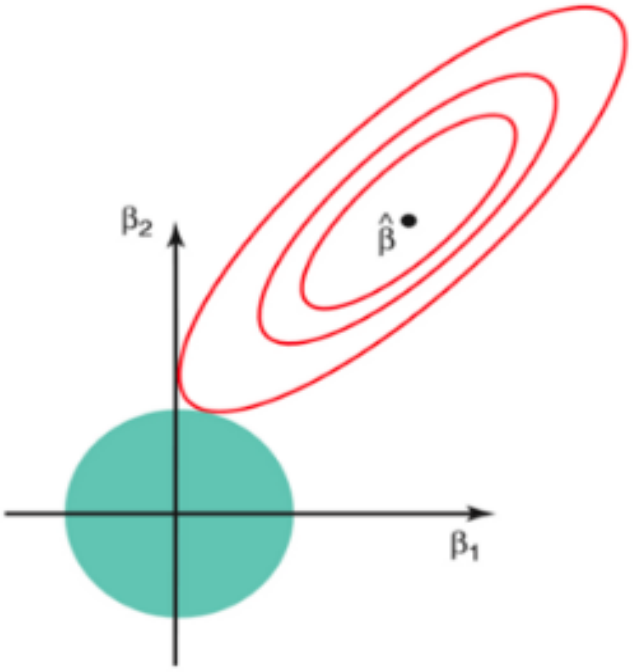

Рисунок 1.4 — Модель рідж регресії

Модель рідж-регресії зображено на Рисунку 1.4. Графік наочно ілюструє обмеження на коефіцієнти моделі у взаємодії із функцією помилки. У цьому випадку точка дотику графіку функції до фігури обмежень коефіцієнтів із меншою вірогідністю буде лежати на осі, що запобігає обнуленню коефіцієнтів.

Інший тип регуляризації – **ласо** чи L1-регуляризація. Замість того аби нараховувати штрафи за кожну ознаку даних, штрафи в ній нараховуються лише за великих значень коефіцієнтів. До того ж ласо здатний обнулювати їх, тим самим повністю видаляючи ознаку з датасету (оскільки при обчисленні результуючої змінної відповідну ознаку буде помножено на нуль). Таким чином, з регресією лассо модель здатна повністю позбутися шумів у даних.

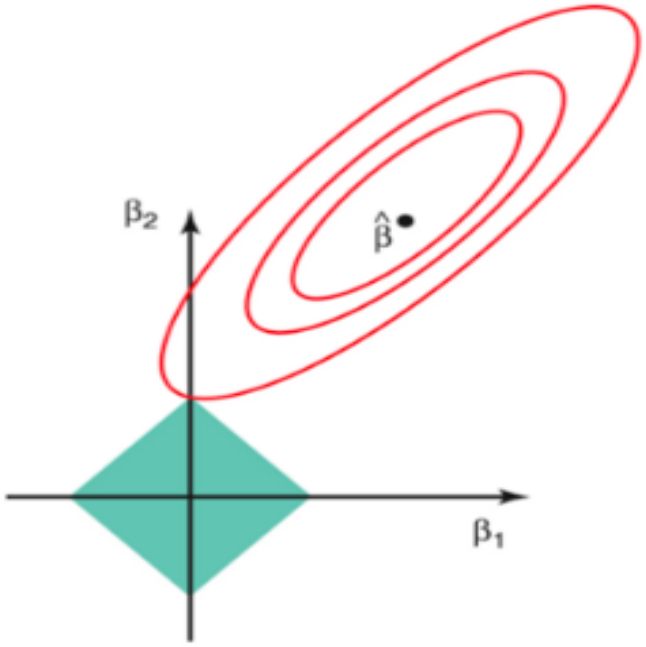

Рисунок 1.5 — Коефіцієнти ласо регресії

Модель ласо-регресії зображено на Рисунку 1.5. Графік наочно ілюструє, що коефіцієнти моделі прийматимуть значення 0, коли перетинатимуть вісь Y.

Генеративно змагальне навчання (ГЗН) – генеративно-змагальна нейронна мережа – один із алгоритмів класичного машинного навчання без учителя. Суть ідеї полягає у комбінації двох нейромереж, при якій одночасно

працюють два алгоритми: “генератор” та “дискримінатор”. Задача генератора – створювати образи заданої категорії, дискримінатора – намагатися розпізнати створений образ. Така настройка нейронної мережі на багатьох прикладах суперництва робить її більш стійкою до атак зловмисників.

“Дискримінаційні алгоритми намагаються класифікувати вхідні дані. Враховуючи особливості отриманих даних, вони намагаються визначити категорію, до якої належать. Натомість, генеративні алгоритми зайняті зворотним. Замість передбачення категорій за наявними образами, вони намагаються підібрати образи до цієї категорії.”[10]

У той час як дискримінаційні алгоритми хвилює взаємозв'язок між y та x, генеративні алгоритми хвилює “звідки беруться ці значення x”. Принцип роботи алгоритму можна описати наступним чином:

1. Генератор отримує рандомне число та повертає зображення.
2. Це згенероване зображення подається дискримінатору разом із потоком зображень, взятих із фактичного набору даних.
3. Дискримінатор приймає як реальні, так і підроблені зображення і повертає числа ймовірності від 0 до 1, причому 1 є справжнім зображенням, в той час як 0 - фальшиве.

Рисунок 1.6 ілюструє модель генеративно змагального навчання.

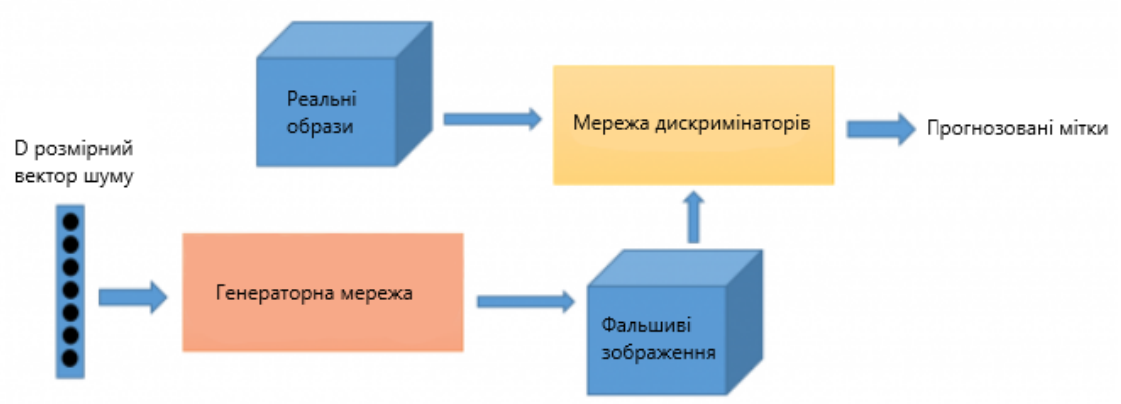

Рисунок 1.6 — Модель генеративно змагального навчання

Змагальне навчання призводить до невеликого зниження точності прогнозів моделі глибокого навчання. Однак таке погіршення вважається прийнятним компромісом із погляду стійкості до атак.

1.3 Формалізована постановка задачі

За результатами літературного огляду нейромережевих технологій аналізу даних було визначено необхідність дослідження нових моделей і методів підвищення робастності до шуму та змагальних атак.

Одним із перспективних напрямків підвищення робастності моделі є підвищення стійкості екстрактора ознакового опису на основі контрастних функцій втрат та їх модифікацій, що збільшують буферну зону між класами розпізнавання.

Дано навчальний набір даних Food 101, що містить 70 розмічених навчальних зразків та архітектуру ResNet-18 як основу для побудови екстрактора ознак.

Метою роботи є підвищення робастності моделі розпізнавання зображень за рахунок зміни алгоритму і протоколу машинного навчання.

Як **критерій робастності** візьмемо відношення точності моделі на зашумлених тестових даних до точності моделі на незашумлених тестових даних. Вимірювання робастності буде здійснюватися на моделі, синтезованій традиційним шляхом, і на моделі, синтезований за запропонованим методом. Результуючі значення запропоновано порівняти за дослідженими критеріями стійкості.

Для досягнення поставленої мети необхідно вирішити ряд задач:

1. Підготовка навчальних даних, а також тестових оригінальних і зашумлених вибірок.
2. Побудова моделі в рамках традиційного підходу з екстрактором ResNet-18 та вихідним повнозв’язним шаром з Softmax нормалізацією.
3. Побудова моделі аналізу даних з використанням модифікації контрастної функції втрат для збільшення буферної зони між класами розпізнавання.

Імплементовані моделі апробовано на тестових оригінальних і зашумлених даних. У результаті проведеного дослідження здійснено порівняльний аналіз отриманих значень стійкості моделей у розрізі процесу їх навчання.

РОЗДІЛ 2. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СИНТЕЗУ РОБАСТНОЇ ДО ШУМУ ГЛИБОКОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Модель і метод навчання екстрактора ознак

Однією із проблем завдання візуальної класифікації та розпізнавання об'єктів є отримання більш дискримінантних ознак для поліпшення та оптимізації роботи глибинних нейронних мереж. Одним з найбільш ефективних і поширених способів вирішення перерахованих вище завдань є використання функції втрат.

Дана функція характеризується втратою при неправильному прийнятті рішень на основі вже отриманих даних. В якості вхідних даних виступають два елементи: вихідне значення поданої моделі та очікуване істине.

У більшості навчальних випадків функція розраховується як різниця між фактичним вихідним значенням y і прогнозованим вихідним значенням ŷ. Функція, використана для обчислення, відома як функція втрат і є ідентифікатором того, наскільки точно модель нейронної мережі здатна прогнозувати очікуваний результат.

Вибір правильної функції втрат має вирішальне значення для навчання точної моделі. У практичних застосуваннях вибір функції витрат буде обмежений багатьма факторами, такими як наявність викидів, вибір алгоритмів машинного навчання, складність у часі градієнтного спуску, складність отримання похідної та достовірність передбаченого значення. Отже, немає однієї функції втрат, придатної для обробки всіх типів даних.

У даній роботі нами запропоновано нову функцію втрат для одночасного розгляду внутрішньокласової компактності та міжкласової роздільності шляхом штрафування за контрастні значення між відстанями

навчальних вибірок до їхніх відповідних центрів класів та сумою відстані навчальних вибірок до їх невідповідних центрів класів (2.1).

$$L_{ct-c} = \frac{1}{2}\sum_{i=1}^{m} \frac{||x_i - c_{y_i}||_2^2}{(\Sigma_{j=1, j\neq y_i}^{k} ||x_i - c_j||_2^2) + \sigma}, \quad (2.1)$$

де L_{ct-c} - втрата контрастного центру,

m - кількість навчальних вибірок у мінімальному пакеті даних,

$x_i \in R_d$ означає і-у навчальну вибірку з розмірністю ознаки *d*,

y_i означає мітку x_i,

$c_{y_i} \in$ Rd означає центр *y*-го класу глибоких особливостей з розмірністю *d*,

k номер класу,

σ - константа, використовувана для того аби знаменник не дорівнював 0.

Таким чином, відмінні ознаки в межах одного класу повинні мати низьку роздільність, у той час як відмінні ознаки різних класів повинні мати високу відокремленість.

Втрата контрастного центру була запропонована для покращення стратегії навчання та отримання більш різноманітних та значущих характеристик за рахунок дослідження не лише внутрішньокласової компактності, але й міжкласового відокремлення. Теоретично, функція втрат була б ідеальною при виконанні задачі класифікації зображень коли існує низька мінливість прикладів за класами та/або висока мінливість прикладів усередині класу.

У даному випадку, в наборі даних Food101 є декілька міжкласових продуктів харчування. Наприклад, деякі зображення розчинної кави дуже

схожі на какао. Більш того, деякі зображення всередині класу також здаються різними. Наприклад, у класі продуктів є декілька зображень не схожих за зовнішніми характеристиками один на одного.

В результаті це дозволяє моделі працювати краще при використанні контрастних втрат у центрі, ніж при використанні перехресних ентропійних втрат.

2.2 Модель класифікатора зображень

Повнозв'язний шар. Повнозв'язний шар є найпростішим і широко застосовуваним шаром нейронної мережі (рис. 2.2). Кожен нейрон в цьому шарі є персептроном з нелінійною функцією активації.

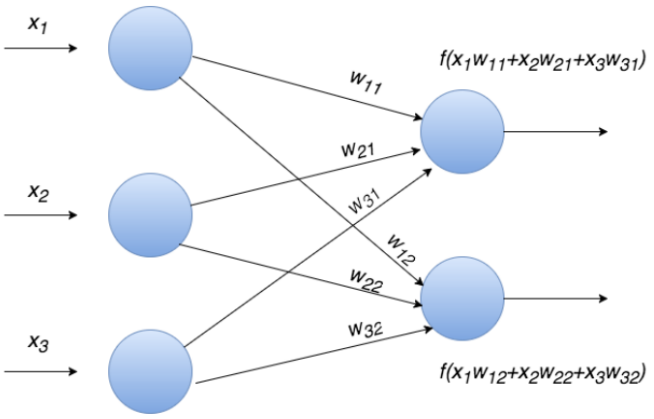

Рисунок 2.2 — Модель повнозв'язного шару

У процесі перетворення даних тривимірна матриця сигналів буде розгорнута у вектор, який буде пропущений через повнозв'язну нейронну мережу. Операція розгортання (рис. 2.3) полягає в "склеюванні" рядків в єдиний числовий ряд. В результаті буде отримано вектор значної величини, який надалі буде перетворено за допомогою багатошарової мережі з повними зв'язками. Опис механізму роботи повнозв'язного шару виглядає наступним

чином: кожен елемент вектора множиться на ваги зв'язків, підсумовуються між собою з деяким зсувом, після чого результат перетворюється за допомогою функції активації.

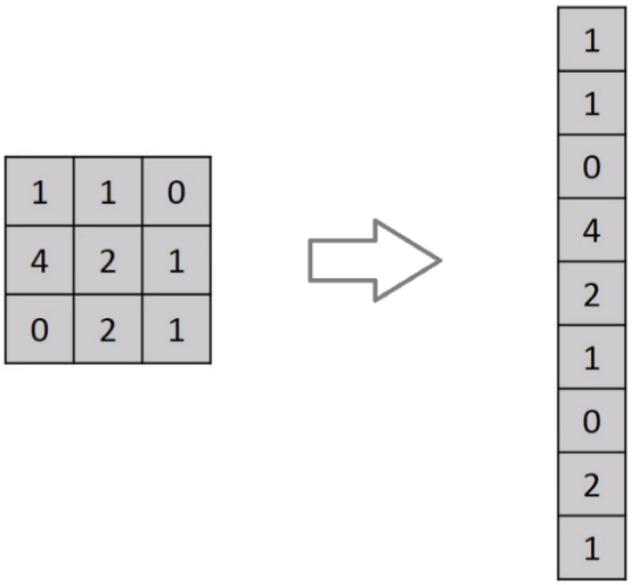

Рисунок 2.3 — Розгортання двовимірної матриці ознак в вектор.

Вихідний шар. Вихідний шар відповідає за формування ймовірностей приналежності вхідного образу тому чи іншому класу (деякому числу). Щоб досягти цього, вихідний шар повинен містити кількість нейронів, які відповідають кількості класів. Зважені та підсумовані сигнали далі модифікуються за допомогою функції активації. У даній роботі нами було використано функцію softmax.

Шар активації - це шар, який застосовує деяку, зазвичай нескладну, функцію до кожної точки вхідного вектора. Існують також функції активації, які залежать від усіх значень відразу, але суть залишається такою самою:

значення вхідного вектора перетворюються на вихідні значення за допомогою деякої функції.

Функцій активації існує чимало. Найбільш відомими є сигмоїдальна, гіперболічний тангенс, ReLU та функція м'якого максимуму (softmax). Функція активації softmax зазвичай застосовується до вихідного шару задач множинної класифікації. Вона може гарантувати, що сума сигналів всіх вихідних нейронів дорівнює 1, а значення кожного виходу належить інтервалу [0,1] і є його ймовірністю. Результатом даної функції є остаточний прогноз. Також Softmax використовується для вирішення мультикласових задач, коли є кілька вихідних нейронів (по одному на клас). Тобто softmax використовується як функція активації для завдань класифікації декількох об'єктів, коли членство потрібне для більш ніж двох позначок класів. Функція Softmax задається наступним чином (2.2):

$$\partial(x_j) = \frac{x^{ej}}{\Sigma_i x^{ej}}. \tag{2.2}$$

Обчислення втрат. Аби визначити, наскільки точно нейронна мережа здатна розпізнавати та класифікувати зашумлені об’єкти на зображеннях, нами буде використано функцію втрат. Результуючі значення функції втрат характеризуватимуть якість нейронної мережі. Зазвичай для оцінки класифікаторів застосовують категоріальну кросентропійну функцію втрат (2.3):

$$H(y,\hat{y}) = \sum_i y_i \, log \, \frac{1}{\hat{y}_i} = -\sum_i y_i \, log \, \hat{y}_i \tag{2.3}$$

де $\hat{y}$ - фактична відповідь нейронної мережі,

y - бажана відповідь нейронної мережі.

Відповіддю у даному випадку буде деяке число, що харектеризує ймовірність приналежності досліджуваного об’єкта до заданих категорій. Для отримання загального показника втрат, притаманного всім категоріям загалом, беруть середнє значення кожної з них.

2.3 Метод зворотнього поширення помилки та оптимізатор Adam

“Алгоритм зворотнього поширення помилки - популярний алгоритм навчання плоскошарових нейронних мереж прямого поширення (багатошарових персептронів). Належить до методів навчання з учителем, тому вимагає, щоб у навчальних прикладах були задані цільові значення”.[5]

“Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилок роботи нейромережевої структури та отримання бажаного виходу. Основна ідея цього методу полягає у поширенні сигналів помилки від виходів мережі до її входів - у напрямку, зворотному прямому поширенню сигналу у звичайному режимі роботи”.[6]

В основі ідеї алгоритму лежить вихідне використання помилки нейронної мережі для обчислення величин корекції вагів нейронів у її прихованих шарах (2.4):

$$\frac{1}{2}\sum_{i=1}^{k}(y - y')^2,$$

(2.4)

де k - число вихідних нейронів мережі,

y - цільове значення,

y' - фактичне вихідне значення.

“Алгоритм є ітеративним і використовує покроковий принцип навчання, коли ваги нейронів мережі коригуються після подачі на її вхід одного навчального прикладу. На кожній ітерації відбувається два проходи мережі: прямий та зворотний. На **прямому** вхідний вектор поширюється від входів мережі до її виходів і формує певний вихідний вектор, що відповідає поточному (фактичному) стану ваги. Потім обчислюється помилка нейронної мережі, як різниця між фактичним та цільовим значеннями”.[11] На зворотному проході ця помилка поширюється від виходу мережі до її входів, і коригуються ваги нейронів відповідно до правила (2.5):

$$\Delta w_{ij}(n) = -\eta \frac{\partial E_{av}}{\partial w_{ij}}, \quad (2.5)$$

де w_{ij} вага і-того зв'язку j-го нейрону,

η - параметр швидкості навчання, що дозволяє керувати величиною кроку корекціїΔw_{ij} .

З метою більш точного налаштування на мінімум помилки η підбирається експериментально у процесі навчання (змінюється в інтервалі від 0 до 1).

“Таким чином, з огляду на вищевказане можемо стверджувати що алгоритм використовує так званий стохастичний градієнтний спуск, «просуваючись» у багатовимірному просторі вагів з метою досягти мінімуму функції помилки. На практиці навчання продовжується не до точного налаштування мережі на мінімум функції помилки, а до тих пір, поки не буде досягнуто досить точного наближення. Це дозволить, з одного боку,

зменшити кількість ітерацій навчання, з другого - уникнути перенавчання мережі”.[7]

Відомо, що найбільш використовуваним методом навчання нейронних мереж є алгоритм зворотного поширення помилки, в якому мінімізація цільової функції проводиться **методом оптимізації Адама**.

Глибинне навчання зачасту вимагає багато часу та комп'ютерних ресурсів для реалізації, що є основним ресурсом для розробки алгоритму глибокого навчання. Хоча ми можемо використовувати розподілене паралельне навчання для прискорення вивчення моделі, необхідні обчислювальні ресурси зовсім не зменшуються. Тільки алгоритм оптимізації, який вимагає менше можливостей, прискорює збіжність моделі. Для цього було створено алгоритм Адама.

Навідміну від підходу стохастичного градієнтного спуску, який підтримує єдину швидкість навчання для всіх оновлень вагів і залишає її не змінною під час тренування, метод Адама ітерує швидкість навчання кожної окремої ваги мережі і адаптується у процесі навчання. Таким чином, метод обчислює індивідуальні адаптивні швидкості навчання для різних параметрів з оцінок першого та другого моментів градієнтів.

Введемо модель методу Адама наступним чином (2.5):

$$\begin{gathered} St := \alpha \cdot St\text{-}1 + (1\text{-}\alpha) \cdot \nabla E^2 t;\ S0 := 0 \\ Dt := \beta \cdot Dt\text{-}1 + (1\text{-}\beta) \cdot \nabla Et;\ D0 := 0 \quad , \\ \Delta Wt := \eta \cdot (Gt + \rho \cdot Wt\text{-}1) + \mu \cdot \Delta Wt\text{-}1 \end{gathered} \tag{2.5}$$

де η- коефіцієнт швидкості тренування,

∇E- градієнт функції втрати,

μ- коефіцієнт моменту.

На даний момент вважається, що даний підхід є одним із найефективніших алгоритмів оптимізації у навчанні нейронних мереж. Адам є популярним алгоритмом у галузі глибинного навчання оскільки він характеризується достатньою швидкістю отримання чітких результатів. Емпіричні дослідження показують, що Адам гарно показує себе на практиці та вигідно відрізняється від інших методів стохастичної оптимізації.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СИНТЕЗУ РОБАСТНОЇ ДО ШУМУ ГЛИБОКОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Формування навчальних та тестових даних

У ході проведеного дослідження було проведено навчання нейронної мережі на вибірці зображень, представлених набором даних Food 101 (рис. 3.1). Даний набір складається зі 101 000 зображень. Зображення набору розділені на 101 клас, такі як: фрукти, овочі, кава, фастфуд та інші. Кожен клас містить 750 навчальних образів та 250 тестових зображень розміром 512x512. Для навчальних зображеннь не було проведено навмисного очищення, тому більшість з них є зашумленими. В основному це проявляється у яскравих кольорах, а іноді - у неправильних етикетках.

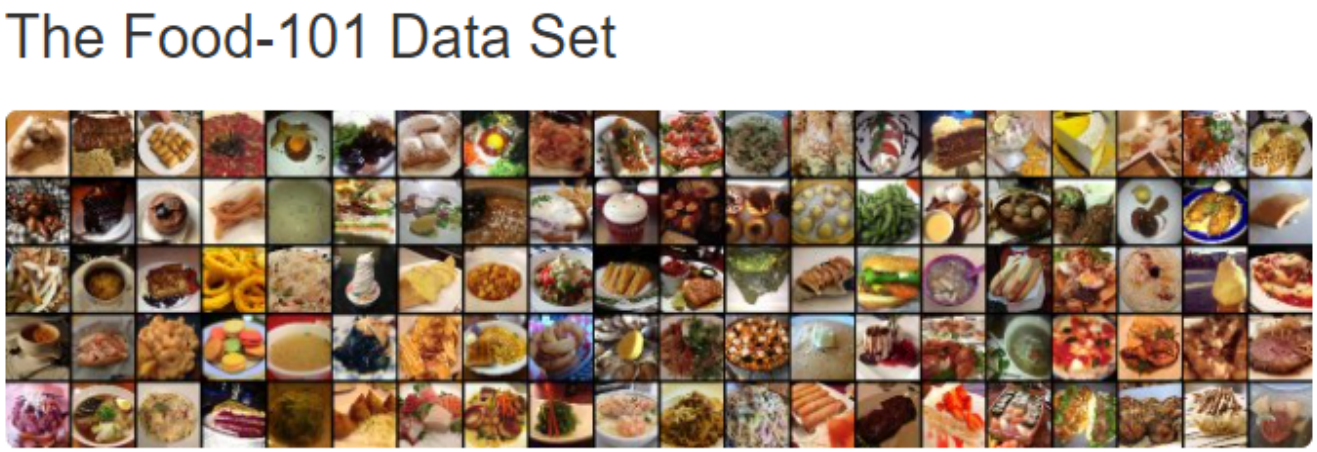

Рисунок 3.1 — Приклади зображень навчальної вибірки

У ході програмної реалізації було імплементовано механізм генерації початкового об’єму даних, на основі якого було проведено наступні ітерації навчання та апробації запропонованих моделей (рис. 3.2).

8.0 Data Transformation Function

In this section, we perform data augmentation for both our training and test data. However, no changes were made with reference to the base model. The same data augmentation was used.

```python
def transfrom_data():
    data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                             std=[0.229, 0.224, 0.225])
        ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                             std=[0.229, 0.224, 0.225])
        ])
    }

    return data_transforms
```

Рисунок 3.2 — Програмна реалізація даних для навчання

Тренувальні зображення харакетризуються достатньою роздільною здатністю. У ході підготовки даних для дослідження запропоновано виділити для кожного елементу вибірки патчі розмірністю 224x224 пікселя як частину початкового зображення у довільному місці.

У межах задачі підготовки даних для дослідження було проведено корекцію контрастності зображень. Для цього елементи вибірки було нормалізовано за допомогою середнього значення для кожного каналу окремо (3.1). Оскільки усереднені значення яскравості розраховані по досить великому датасету, результат ділиться на значення дисперсії (3.1).

transforms.Normalize (mean=[0.485, 0.456, 0.406],

(3.1)

std=[0.229, 0.224, 0.225])

3.2 Короткий опис програмного забезпечення

У ході роботи було проведено практичну реалізацію нейромережевої структури для розпізнавання об'єктів з підвищеним рівнем робастності до шуму за допомогою мови програмування Python.

Стандартний процес обробки даних був виконаний за допомогою класифікатора і складався з наступних кроків:

1. фільтрація та інтеграція даних;
2. нормалізація елементів вибірки;
3. аналіз даних і оцінка шаблонів.

Програмування нейронної мережі полягало у реалізації програмного комплексу для виконання задач, які описані нижче.

Вилучення даних. На даному етапі було сформовано колекції тренувальних та навчальних даних за допомогою датасету Food-101.

```
def extract_data(filename, num_images, IMAGE_WIDTH):
    '''
    Extract images by reading the file bytestream. Reshape the read values into a 3D matrix of dimensions [m, h, w], where m
    is the number of training examples.
    '''
    print('Extracting', filename)
    with gzip.open(filename) as bytestream:
        bytestream.read(16)
        buf = bytestream.read(IMAGE_WIDTH * IMAGE_WIDTH * num_images)
        data = np.frombuffer(buf, dtype=np.uint8).astype(np.float32)
        data = data.reshape(num_images, IMAGE_WIDTH*IMAGE_WIDTH)
        return data

def extract_labels(filename, num_images):
    '''
    Extract label into vector of integer values of dimensions [m, 1], where m is the number of images.
    '''
    print('Extracting', filename)
    with gzip.open(filename) as bytestream:
        bytestream.read(8)
        buf = bytestream.read(1 * num_images)
        labels = np.frombuffer(buf, dtype=np.uint8).astype(np.int64)
    return labels
```

Рисунок 3.3 — Вилучення тренувальних даних та відповідних їм маркерів

Файли цієї колекції зберігають зображення та відповідні маркери у вигляді багатовимірних масивів. Завдяки цьому було значно спрощено роботу з даними шляхом уникнення процесу збору, очищення та інтеграції випадкових реальних зображень. Для отримання даних із файлів було імплементовано та застосовано окремі функції (рис. 3.3).

Ініціалізація параметрів. Наступним кроком було реалізовано функції для ініціалізації параметрів фільтрів для згорткових шарів та вагових коефіцієнтів повнозв'язних шарів. Аби процес навчання був рівномірним, нами проведено ініціалізацію, в якій середнє значення параметрів дорівнюватиме нулю, а стандартне відхилення - одиниці (рис. 3.4).

```
def initializeFilter(size, scale = 1.0):
    '''
    Initialize filter using a normal distribution with and a
    standard deviation inversely proportional the square root of the number of units
    '''
    stddev = scale/np.sqrt(np.prod(size))
    return np.random.normal(loc = 0, scale = stddev, size = size)

def initializeWeight(size):
    '''
    Initialize weights with a random normal distribution
    '''
    return np.random.standard_normal(size=size) * 0.01
```

Рисунок 3.4 — Функції ініціалізації параметрів нейронної мережі

Зворотне поширення помилки. Для навчання нейронної мережі нами було реалізовано функції для зворотного поширення помилки через пулінгові та згорткові шари (рис. 3.5). Застосований підхід, в свою чергу, дозволив провести ітеративне коригування ваг мережі.

```
def convolutionBackward(dconv_prev, conv_in, filt, s):
    '''
    Backpropagation through a convolutional layer.
    '''
    (n_f, n_c, f, _) = filt.shape
    (_, orig_dim, _) = conv_in.shape
    ## initialize derivatives
    dout = np.zeros(conv_in.shape)
    dfilt = np.zeros(filt.shape)
    dbias = np.zeros((n_f,1))
    for curr_f in range(n_f):
        # loop through all filters
        curr_y = out_y = 0
        while curr_y + f <= orig_dim:
            curr_x = out_x = 0
            while curr_x + f <= orig_dim:
                # loss gradient of filter (used to update the filter)
                dfilt[curr_f] += dconv_prev[curr_f, out_y, out_x] * conv_in[:, curr_y:curr_y+f, curr_x:curr_x+f]
                # loss gradient of the input to the convolution operation (conv1 in the case of this network)
                dout[:, curr_y:curr_y+f, curr_x:curr_x+f] += dconv_prev[curr_f, out_y, out_x] * filt[curr_f]
                curr_x += s
                out_x += 1
            curr_y += s
            out_y += 1
        # loss gradient of the bias
        dbias[curr_f] = np.sum(dconv_prev[curr_f])

    return dout, dfilt, dbias

def nanargmax(arr):
    '''
    return index of the largest non-nan value in the array. Output is an ordered pair tuple
    '''
    idx = np.nanargmax(arr)
    idxs = np.unravel_index(idx, arr.shape)
    return idxs

def maxpoolBackward(dpool, orig, f, s):
    '''
    Backpropagation through a maxpooling layer. The gradients are passed through the indices of greatest value in the original maxpooling during the forward step.
    '''
    (n_c, orig_dim, _) = orig.shape

    dout = np.zeros(orig.shape)

    for curr_c in range(n_c):
        curr_y = out_y = 0
        while curr_y + f <= orig_dim:
            curr_x = out_x = 0
            while curr_x + f <= orig_dim:
                # obtain index of largest value in input for current window
                (a, b) = nanargmax(orig[curr_c, curr_y:curr_y+f, curr_x:curr_x+f])
                dout[curr_c, curr_y+a, curr_x+b] = dpool[curr_c, out_y, out_x]

                curr_x += s
                out_x += 1
            curr_y += s
            out_y += 1

    return dout
```

Рисунок 3.5 — Функція зворотного поширення градієнта через згортковий та пулінговий шари

Побудова мережі. На даному етапі нами застосовано функцію, яка комбінує в собі операції прямого та зворотного поширення сигналів у згорткових шарах. Ця функція приймає вхідний образ. Її результучим значенням є градієнти та характеристика втрат.

```
def conv(image, label, params, conv_s, pool_f, pool_s):

    [f1, f2, w3, w4, b1, b2, b3, b4] = params

    ################################################
    ############## Forward Operation ###############
    ################################################
    conv1 = convolution(image, f1, b1, conv_s) # convolution operation
    conv1[conv1<=0] = 0 # pass through ReLU non-linearity

    conv2 = convolution(conv1, f2, b2, conv_s) # second convolution operation
    conv2[conv2<=0] = 0 # pass through ReLU non-linearity

    pooled = maxpool(conv2, pool_f, pool_s) # maxpooling operation

    (nf2, dim2, _) = pooled.shape
    fc = pooled.reshape((nf2 * dim2 * dim2, 1)) # flatten pooled layer

    z = w3.dot(fc) + b3 # first dense layer
    z[z<=0] = 0 # pass through ReLU non-linearity

    out = w4.dot(z) + b4 # second dense layer

    probs = softmax(out) # predict class probabilities with the softmax activation function

    ################################################
    ################## Loss ########################
    ################################################

    loss = categoricalCrossEntropy(probs, label) # categorical cross-entropy loss

    ################################################
    ############# Backward Operation ###############
    ################################################
    dout = probs - label # derivative of loss w.r.t. final dense layer output
    dw4 = dout.dot(z.T) # loss gradient of final dense layer weights
    db4 = np.sum(dout, axis = 1).reshape(b4.shape) # loss gradient of final dense layer biases

    dz = w4.T.dot(dout) # loss gradient of first dense layer outputs
    dz[z<=0] = 0 # backpropagate through ReLU
    dw3 = dz.dot(fc.T)
    db3 = np.sum(dz, axis = 1).reshape(b3.shape)

    dfc = w3.T.dot(dz) # loss gradients of fully-connected layer (pooling layer)
    dpool = dfc.reshape(pooled.shape) # reshape fully connected into dimensions of pooling layer

    dconv2 = maxpoolBackward(dpool, conv2, pool_f, pool_s) # backprop through the max-pooling layer(only neurons with highest activation in window get updated)
    dconv2[conv2<=0] = 0 # backpropagate through ReLU

    dconv1, df2, db2 = convolutionBackward(dconv2, conv1, f2, conv_s) # backpropagate previous gradient through second convolutional layer.
    dconv1[conv1<=0] = 0 # backpropagate through ReLU

    dimage, df1, db1 = convolutionBackward(dconv1, image, f1, conv_s) # backpropagate previous gradient through first convolutional layer.

    grads = [df1, df2, dw3, dw4, db1, db2, db3, db4]

    return grads, loss
```

Рисунок 3.6 — Функція зворотного поширення градієнта через згортковий та пулінговий шари

Навчання нейронної мережі. Аби нейронна мережа мала змогу навчатися вирішенню класифікаційних задач, нами було застосовано алгоритм оптимізації Adam (рис. 3.7).

```
def adamGD(batch, num_classes, lr, dim, n_c, beta1, beta2, params, cost):
    '''
    update the parameters through Adam gradient descnet.
    '''
    [f1, f2, w3, w4, b1, b2, b3, b4] = params

    X = batch[:,0:-1] # get batch inputs
    X = X.reshape(len(batch), n_c, dim, dim)
    Y = batch[:,-1] # get batch labels

    cost_ = 0
    batch_size = len(batch)

    # initialize gradients and momentum,RMS params
    df1 = np.zeros(f1.shape)
    df2 = np.zeros(f2.shape)
    dw3 = np.zeros(w3.shape)
    dw4 = np.zeros(w4.shape)
    db1 = np.zeros(b1.shape)
    db2 = np.zeros(b2.shape)
    db3 = np.zeros(b3.shape)
    db4 = np.zeros(b4.shape)

    v1 = np.zeros(f1.shape)
    v2 = np.zeros(f2.shape)
    v3 = np.zeros(w3.shape)
    v4 = np.zeros(w4.shape)
    bv1 = np.zeros(b1.shape)
    bv2 = np.zeros(b2.shape)
    bv3 = np.zeros(b3.shape)
    bv4 = np.zeros(b4.shape)

    s1 = np.zeros(f1.shape)
    s2 = np.zeros(f2.shape)
    s3 = np.zeros(w3.shape)
    s4 = np.zeros(w4.shape)
    bs1 = np.zeros(b1.shape)
    bs2 = np.zeros(b2.shape)
    bs3 = np.zeros(b3.shape)
    bs4 = np.zeros(b4.shape)

    for i in range(batch_size):

        x = X[i]
        y = np.eye(num_classes)[int(Y[i])].reshape(num_classes, 1) # convert label to one-hot

        # Collect Gradients for training example
        grads, loss = conv(x, y, params, 1, 2, 2)
        [df1_, df2_, dw3_, dw4_, db1_, db2_, db3_, db4_] = grads

        df1+=df1_
        db1+=db1_
        df2+=df2_
        db2+=db2_
        dw3+=dw3_
        db3+=db3_
        dw4+=dw4_
        db4+=db4_

        cost_+= loss

    # Parameter Update

    v1 = beta1*v1 + (1-beta1)*df1/batch_size # momentum update
    s1 = beta2*s1 + (1-beta2)*(df1/batch_size)**2 # RMSProp update
    f1 -= lr * v1/np.sqrt(s1+1e-7) # combine momentum and RMSProp to perform update with Adam

    bv1 = beta1*bv1 + (1-beta1)*db1/batch_size
    bs1 = beta2*bs1 + (1-beta2)*(db1/batch_size)**2
    b1 -= lr * bv1/np.sqrt(bs1+1e-7)
```

```
    v2 = beta1*v2 + (1-beta1)*df2/batch_size
    s2 = beta2*s2 + (1-beta2)*(df2/batch_size)**2
    f2 -= lr * v2/np.sqrt(s2+1e-7)

    bv2 = beta1*bv2 + (1-beta1) * db2/batch_size
    bs2 = beta2*bs2 + (1-beta2)*(db2/batch_size)**2
    b2 -= lr * bv2/np.sqrt(bs2+1e-7)

    v3 = beta1*v3 + (1-beta1) * dw3/batch_size
    s3 = beta2*s3 + (1-beta2)*(dw3/batch_size)**2
    w3 -= lr * v3/np.sqrt(s3+1e-7)

    bv3 = beta1*bv3 + (1-beta1) * db3/batch_size
    bs3 = beta2*bs3 + (1-beta2)*(db3/batch_size)**2
    b3 -= lr * bv3/np.sqrt(bs3+1e-7)

    v4 = beta1*v4 + (1-beta1) * dw4/batch_size
    s4 = beta2*s4 + (1-beta2)*(dw4/batch_size)**2
    w4 -= lr * v4 / np.sqrt(s4+1e-7)

    bv4 = beta1*bv4 + (1-beta1)*db4/batch_size
    bs4 = beta2*bs4 + (1-beta2)*(db4/batch_size)**2
    b4 -= lr * bv4 / np.sqrt(bs4+1e-7)


    cost_ = cost_/batch_size
    cost.append(cost_)

    params = [f1, f2, w3, w4, b1, b2, b3, b4]

    return params, cost

#####################################################
################## Training #########################
#####################################################

def train(num_classes = 10, lr = 0.01, beta1 = 0.95, beta2 = 0.99, img_dim = 28, img_depth = 1, f = 5, num_filt1 = 8, num_filt2 = 8, batch_size = 32, num_epochs = 2, save_path = 'params.pkl'):

    # Get training data
    m =50000
    X = extract_data('train-images-idx3-ubyte.gz', m, img_dim)
    y_dash = extract_labels('train-labels-idx1-ubyte.gz', m).reshape(m,1)
    X-= int(np.mean(X))
    X/= int(np.std(X))
    train_data = np.hstack((X,y_dash))

    np.random.shuffle(train_data)

    ## Initializing all the parameters
    f1, f2, w3, w4 = (num_filt1 ,img_depth,f,f), (num_filt2 ,num_filt1,f,f), (128,800), (10, 128)
    f1 = initializeFilter(f1)
    f2 = initializeFilter(f2)
    w3 = initializeWeight(w3)
    w4 = initializeWeight(w4)

    b1 = np.zeros((f1.shape[0],1))
    b2 = np.zeros((f2.shape[0],1))
    b3 = np.zeros((w3.shape[0],1))
    b4 = np.zeros((w4.shape[0],1))

    params = [f1, f2, w3, w4, b1, b2, b3, b4]

    cost = []

    print("LR:"+str(lr)+", Batch Size:"+str(batch_size))

    for epoch in range(num_epochs):
        np.random.shuffle(train_data)
        batches = [train_data[k:k + batch_size] for k in range(0, train_data.shape[0], batch_size)]

        t = tqdm(batches)
        for x,batch in enumerate(t):
            params, cost = adamGD(batch, num_classes, lr, img_dim, img_depth, beta1, beta2, params, cost)
            t.set_description("Cost: %.2f" % (cost[-1]))


    with open(save_path, 'wb') as file:
        pickle.dump(params, file)

    return cost
```

Рисунок 3.7 — Програмна реалізація алгоритму оптимізації Adam

Також у ході реалізації нейронної мережі було використано низку бібліотек що приведені в Таблиці 3.1:

Таблиця 3.1 - *Перелік бібліотек використаних у ході роботи*

Назва бібліотеки	**Опис**
Keras	Keras — бібліотека для глибинного вивчення Python, рекомендована для використання початківцям.

Продовження таблиці 3.1 - *Перелік бібліотек використаних у ході роботи*

Назва бібліотеки	**Опис**
	Його мінімалістичний, модульний підхід дозволяє досить легко будувати і запускати глибокі нейронні мережі. Вона сумісна з TensorFlow, Theano, Microsoft Cognitive та ін. Серед особливостей Keras можна виділити наступні характеристики. **Зручність у використанні.** Keras був розроблений, в першу чергу, для користувачів саме тому використовує спеціальні методи: мінімізує кількість дій користувача, необхідних для вирішення поширених задач. **Модульність**. У даному випадку ми маємо послідовність або граф автономних, повністю налаштованих модулів, які можуть бути підключені без додаткових обмежень. **Розширюваність**. Можливість Keras легко додавати нові класи, модулі та функції робить її відмінним засобом для проведення різноманітних досліджень.
tensorflow	TensorFlow – бібліотека для машинного навчання, розроблена компанією Google для побудови та тренування нейронних мереж.

Продовження таблиці 3.1 - *Перелік бібліотек використаних у ході роботи*

Назва бібліотеки	**Опис**
	Він чудово підходить для автоматичного знаходження та

	класифікації образів, оскільки якість розпізнавання наближається до людського сприйняття. TensorFlow може працювати у паралельному режимі на кількох процесора таких як: CPU, і GPU. Для обчислень загального призначення в графічних процесорах використовується CUDA. Це забезпечує високу швидкість навчання та роботи навчених моделей.
Theano	“Theano – це розширення мови Python, що дозволяє ефективно обчислювати математичні вирази, що містять багатовимірні масиви. Бібліотека реалізована мовою Python та підтримується на операційних системах Windows, Linux та Mac OS. До складу Theano входить компілятор, який перекладає математичні вирази, написані мовою Python в ефективний код C або CUDA. Theano надає базовий набір інструментів конфігурації нейромереж та їх навчання.” [8]
NumPy	“NumPy - це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом із великою бібліотекою високорівневих (і дуже швидких) математичних функцій для операцій із цими масивами. Основним об'єктом NumPy є масив numpy.ndarray, що є багатовимірним масивом елементів одного типу”.[9]

3.3 Результати машинного навчання

У результаті проведеного дослідження було практично реалізовано програмний комплекс на основі моделі нейронної мережі згорткового типу Resnet 18. Імплементована система запропонована для покращення ефективності задач розпізнавання об’єктів на зображеннях, так як характеризується підвищенним рівнем робастності до шуму.

У процесі навчання нейромережі було використано вибірку зображеннь, представленої набором даних Food-101, що містить 101 клас зображень, представлений 750 навчальними та 250 текстовими зображеннями розміром 512x512. Початковий набір даних був відповідним чином підготовлений та нормалізований задля оптимізації подальшого використання алгоритмом.

У межах чисельного експерименту було проведено навчення мережі традиційним методом типу з кінця в кінець. Результати роботи алгоритму (рис. 3.8) на навчальному та тестовому наборах даних представлено у вигляді графіків динаміки досліджуваного параметру точності моделі (по осі ординат) у розрізі відповідних ітерацій епох навчання (по осі абцис).

Для дослідження та аналізу якості моделі було проведено перенавчання мережі Resnet 18. Для цього вхідні елементи тестової вибірки були змінені шляхом накладання Гаусового шуму. Для апробації моделі до зображень був застосований шум рівня (PSNR) 30%.

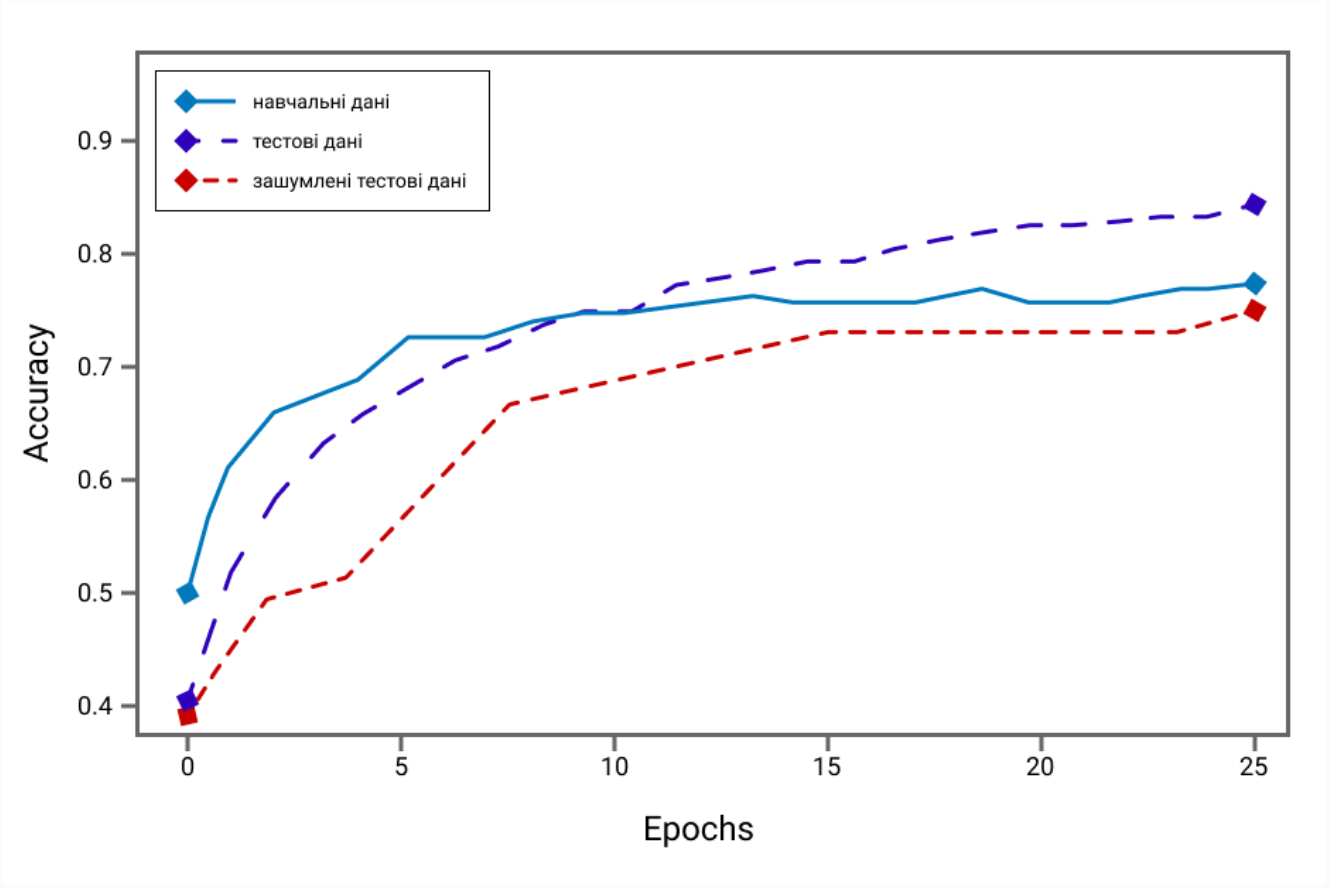

Рисунок 3.8 — Крива зміни точності моделі, навченої традиційним методом.

Аналіз результатів чисельного експерименту навчання нейронної мережі (рис. 3.8) показує неоптимальність роботи моделі навченої традиційним методом. Це виражається у значному падінні точності розпізнавання об’єктів для вибірки із зашумленням, фінальне значення якого (для останньої епохи) становить 0.73 у порівнянні із 0.83 для навчальних і 0.78 для тестових даних. Саме це ілюструє змешення точності валідаційної кривої на 5%.

Після отримання результуючих значень точності моделі, навченою традиційним методом, пропонується провести повторний експеримент на іншій моделі. Для цього для сформованих вибірок даних (навчальних, тестових та зашумлених тестових) проведено навчання екстрактору ознак з

контрастно-центрованою функцією втрат. У результаті проведеного дослідження що доповнює екроентропійну функцію втрат для класифікатора екстрактора ознак це має збільшити буферну зону між класами в просторі ознак.

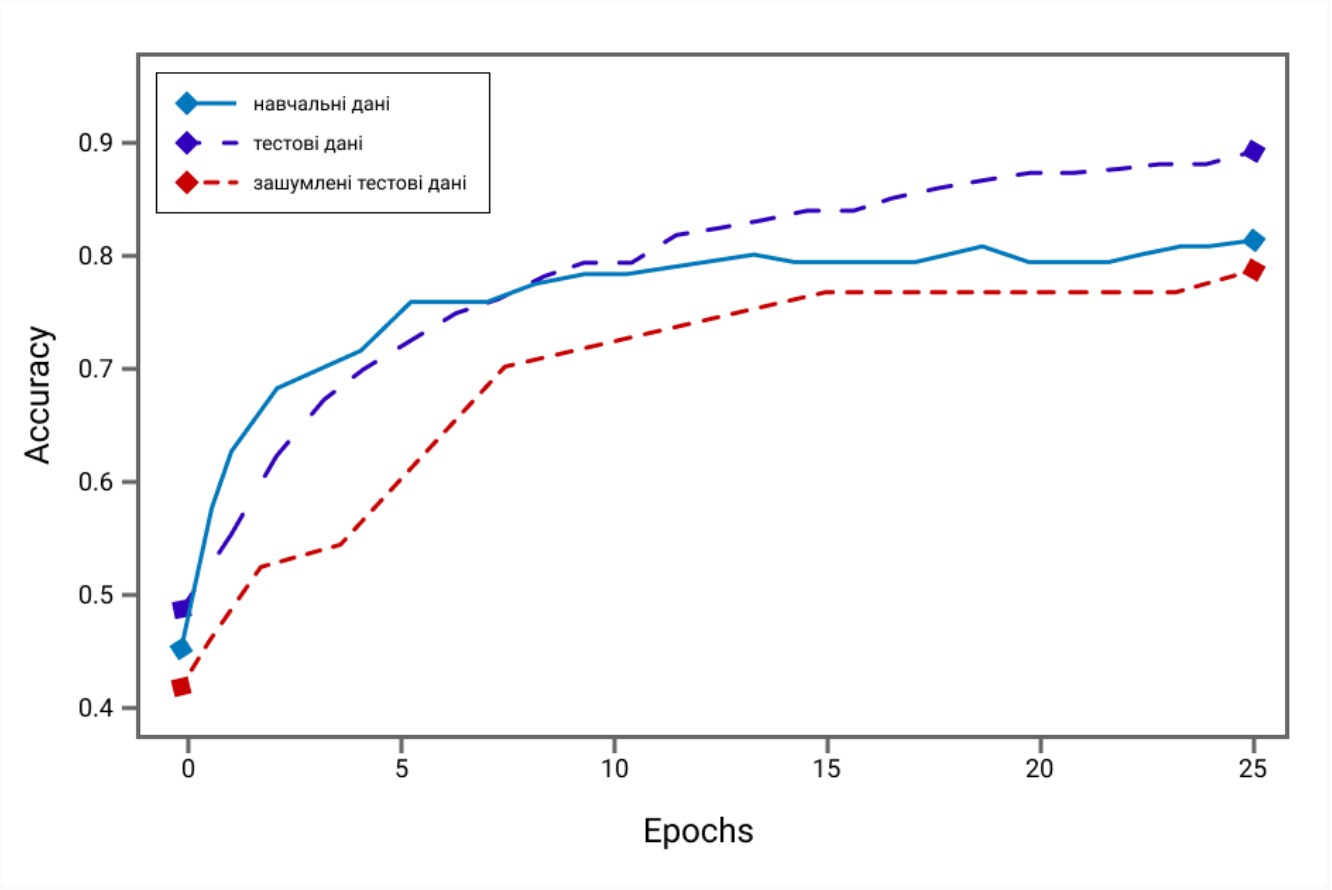

Рисунок 3.9 — Крива зміни точності моделі з контрастно-центрованою функцією втрат

Результати чисельного експерименту наочно ілюструють покращення точності отриманих результатів на 2х вибірках дослідження: 0.89 проти 0.83 для тестових даних; 0.795 проти 0.78 для зашумлених тестових. Показник точності навчання на навчальних даних не показав помітних змін.

Апробація побудованих моделей та порівняльний аналіз отриманих показників якості навчання дозволяє зробити висновок, що найкращі

показники точності та ефективності у межах задачі розпізнавання зображень можна отримати із застосуванням методу з використанням контрастно центрованої функції втрат. Таким чином, експериментально було досліджено та підтверджено що ефективність використання контрастно-центрованої функції втрат для підвищення робастності моделі аналізу даних.

ВИСНОВОК

Штучні нейронні мережі набули поширення в усіх галузях господарства, оскільки їх здатність моделювати когнітивні механізми людини, дозволяє підвищити рівень автоматизації і економічну ефективність багатьох процесів. Проте досі актуальною задачею залишається підвищення робастності моделей нейромереж до шуму і змагальних атак без значного підвищення надлишковості їх структури.

Особливо чутливим до шуму та змагальних атак є згорткові мережі для розпізнавання зображень, тому в межах даного дослідження було запропоновано використати модифікацію алгоритму навчання, що спрямований на збільшення буферних зон між класами в просторі ознак.

У ході дослідження відповідно до поставлених задач, було здійснено програмну реалізацію запропонованої моделі і отримано наступні наукові результати:

1. З метою підбору найоптимальнішого варіанту для підвищення робастності моделей розпізнавання зображень - проаналізовано існуючі підходи до навчання моделі екстрактора ознак та визначено оптимальну модель.
2. У межах задачі перевірки гіпотези про підвищення робастності моделі до шуму, було проведено низку чисельних експериментів з моделлю, що навчена традиційним методом, та з моделлю, де екстрактор ознак навчався з контрастно-центрованою функцією втрат; у ході порівняльного аналізу отриманих результатів

виявлено чисельні характеристики параметрів якості навчання для кожної вибірки даних у розрізі окремої досліджуваної моделі.

3. Експериментально підтверджено, що використання контрастно-центрованої функції втрат дозволяє підвищити точність моделі за умов шумового впливу, тобто забезпечує підвищення робастності порівняно з традиційним методом навчання.

Експериментальні дослідження були проведені на наборі даних Food 101 та нейромережевій архітектурі ResNet-18.

Мережу було навчено на основі тестових даних і здійснено підбір вхідних даних для класифікації на основі датасету Food 101. Даний набір містить 101 клас, кожен з яких складається із 750 навчальних образів та 250 тестових зображень розміром 512x512. Для навчальних зображеннь не було проведено навмисного очищення, тому більшість з них являються зашумленими.

У межах чисельного експерименту було проведено навчання мережі для класифікатора екстрактора ознак з контрастно-центрованою функцією втрат, для цього вхідні елементи тестової вибірки були змінені шляхом накладання Гаусового шуму. Для апробації моделі до зображень був застосований шум рівня (PSNR) 30%. Результати чисельного експерименту наочно ілюструють покращення точності отриманих результатів на 2х вибірках дослідження: 0.89 проти 0.83 для тестових даних; 0.795 проти 0.78 для зашумлених тестових. Показник точності навчання на навчальних даних не показав помітних змін.

За результатам проведених експериментів було підсумовано, що найкращим із запропонованих методів розпізнавання зображень за точністю і

ефективністю слід вважати метод з використанням контрастно центрованої функції втрат.

Список літератури:

1. Чичварин, Н. В. Распознавание образов / Н. В. Чичварин [Электронный ресурс] / Национальная библиотека им. Н. Э. Баумана.

2. Історія розвитку ансамблевих методів класифікації в машинному навчанні https://www.researchgate.net/publication/278019662_Istoria_razvitia_ansamblevyh_metodov_klassifikacii_v_masinnom_obucenii

3. Регрессия в машинном обучении: оптимальный алгоритм . URL: https://proglib.io/p/ml-regression/ (Дата звернення: 6.05.2020)

4. Ансамблеві методи: беггінг, бустінг і стекінг https://neurohive.io/ru/osnovy-data-science/ansamblevye-metody-begging-busting-i-steking/

5. Simon Haykin. Neural Networks: A Comprehensive Foundation(2nd edition). McMaster University, Ontario Canada, 1999. 1088 p.

6. *Барцев С. И., Гилев С. Е., Охонин В. А.*, Принцип двойственности в организации адаптивных сетей обработки информации

7. Kohavi R. Wrappers for feature subset selection, Artificial Intelligence. Volume 97, Issues 1–2, 1997, Pages 273-324.

8. Порівняння бібліотек глибокого навчання [Електронний ресурс] – Режим доступу до ресурсу: https://habrahabr.ru/company/intel/blog/254747/

9. Yann LeCun Leon Bottou, Y. B. Gradient-based learning applied to document recognition / Yoshua Bengio Yann LeCun, Leon Bottou, Patrick Haffner

10. Путятін, Є. П., Гороховатський, В. О., & Матат, О. О. (2006). Методи та алгоритми комп’ютерного зору: навч. посіб. Харків: ТОВ «Компанія СМІТ».

11. Система для аналізу великих масивів даних за допомогою алгоритмів машинного навчання Пронін С. В., Мірошниченко М. О