Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів комп'ютерного зору для вирішення задач навчання ознак для реідентифікації об'єктів
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-3
Танасюк Д.О.
(прізвище, ініціали)

спеціальності 121– Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-наукової програми
(тип програми)
Інженерія програмного забезпечення
(тип програми)

Керівник к.т.н., доц. Турута О.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

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

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Танасюку Дмитру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту): Дослідження методів комп'ютерного зору для вирішення задач навчання ознак для реідентифікації об'єктів
затверджена наказом по університету від "27" березня 2020 р. №473 Ст
2. Термін подання студентом роботи (проекту)
15 травня 2020 р.
3. Вихідні дані до роботи (проекту) модулі для аналізу наборів даних, модулі для навчання нейронних мереж, отримання та візуалізації прогнозу з нейронних моделей, пояснювальна записка. Використовувати ОС Windows, мову програмування Python.

4. Перелік питань, що потрібно опрацювати в роботі: мета роботи, аналіз проблемної галузі і постановка задачі, огляд існуючих методів з реідентифікації об'єктів на зображенні, застосування методів для поставленої задачі, аналіз якості моделей навчання ознак, пошук способів підвищення якості та аналіз їх ефективності.

(Наступний аркуш завдання)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	03 квітня 2020 р.	
2.	Огляд існуючих методів	10 квітня 2020 р.	
3.	Дослідження методів класифікації томографічних зображень	17 квітня 2020 р.	
4.	Підготовка пояснювальної записки	24 квітня 2020 р.	
5.	Спецчастина	29 квітня 2020 р.	
6.	Підготовка презентації та доповіді	30 квітня 2020 р.	
7.	Попередній захист	05 травня 2020 р.	
8.	Нормоконтроль, рецензування	11 травня 2020 р.	
9.	Занесення диплома в електронний архів	15 травня 2020 р.	
10.	Допуск до захисту у зав. кафедри	16 травня 2020 р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання 27 березня 2020 р.

Студент_____
(підпис)

Керівник роботи (проекту) _____ к.т.н., доцент Турута О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до роботи містить: 79 с., 32 рис., 5 табл., 20 формул, 34 джерела.

НАВЧАННЯ ОЗНАК, КОМП'ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, РЕІНДЕТИФІКАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, СІАМСЬКА НЕЙРОННА МЕРЕЖА, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА

Метою роботи є дослідження методів навчання ознак для вирішення задач реідентифікації об'єктів на зображеннях, аналіз результатів їх роботи та шляхів покращення існуючих моделей.

У результаті роботи було оглянуто наукову та патентну літературу за темою, проаналізовано стан розв'язання проблеми за матеріалами публікацій, були розглянуті поняття навчання ознак, описано декілька найпопулярніших моделей та запропоновано модифікацію для підвищення якості реідентифікації.

METRIC LEARNING, COMPUTER VISION, MACHINE LEARNING, RE-IDENTIFICATION, ARTIFICIAL INTELLIGENCE, SIAMIC NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORK

The aim of this work is to research metric learning methods for solving object re-identification problems, analysis of the results of their work and ways to improve existing models.

As a result, the scientific and patent literature on the topic was reviewed, the state of problem solving was analyzed based on publications, the concepts of metric learning were considered, some of the most popular models were described and a modification was proposed to improve the quality of object reidentification.

ЗМІСТ

ВСТУП

Штучний інтелект набуває все більшої популярності. Комп’ютерний зір, один з напрямів штучного інтелекту, використовується у багатьох сферах, від розпізнання області обличчя для кращої якості фото, до сфер безпеки життя – контрольних пунктів чи камер відеоспостереження. Якість (точність) роботи алгоритмів розпізнавання та реідентифікації – одна з головних метрик забезпечення безпеки при використанні комп’ютерного зору.

Комп’ютерний зір – це напрямок в області штучного інтелекту, та технології отримання зображень об'єктів, що з ним пов’язані, а також обробки цих зображень та використання даних, що отримано, для вирішення прикладних задач без участі людини. Завдяки комп’ютерному зору можна вирішити такі задачі, як сегментація та класифікація зображень, розпізнавання обличь, генерація зображень, тощо. Комп’ютерний зір використовується для роботи безпілотних автомобілів, розпізнавання зображень с камер стеження для подальшого використання (наприклад розпізнавання номерів машин, обличь), сортування чи пошук браку на виробництві, картографічні системи, аналіз емоційного стану людини, зчитування штрих-кодів, технології доповненої та віртуальної реальності, конвертація книг та документів із паперового формату до цифрового.

Проектування систем штучного інтелекту, а також розробка програмного забезпечення для систем відео контролю – одні з напрямків, за яким виконуються дослідження кафедри Програмної інженерії.

Метою даної роботи є дослідження, аналіз та покращення якості роботи алгоритмів, що використовуються для реідентифікації об’єктів на зображеннях.

Об’єктом дослідження є архітектури глибоких нейронних згорткових мереж для реідентифікації об’єктів на зображеннях за допомогою методів навчання ознак.

Предметом дослідження є оптимізація метрик якості на різних підзадачах реідентифікації.

На теоретичному рівні дослідження буде проведено аналіз і синтез напрямків покращення метрик якості розпізнавання об’єктів, на емпіричному рівні будуть використовуватися методи експерименту та порівняння для знаходження найоптимальніших параметрів та методів для модифікації алгоритмів реідентифікації, що покращить якість роботи глибоких нейронних мереж навчання ознак.

Дана робота є удосконаленням існуючої архітектури нейронної мережі навчання ознак ArcFace, що являє собою state of the art підходом на більшості задач реідентифікації та багатокласової класифікації.

Тези на тему магістерської роботи було опубліковано на конференції «SCIENCE, RESEARCH, DEVELOPMENT #28. TECHNICS AND TECHNOLOGY» 29.04.2020-30.04.2020 в Баку.

Практична цінність робота полягає у тому, що одержаний за допомогою запропонованої архітектури приріст якості реідентифікації є дуже критичним. Ця технологія зазвичай використовується у системах безпеки, системах відеоспостереження та системах трекінга, тож від якості ідентифікації залежить безпека людей та їхніх даних.

1 АНАЛІЗ СТАНУ ДОСЛІДЖЕНЬ

Останнім часом прогрес у розвитку глибоких згорткових нейронних мереж значно покращив продуктивність широкого спектру завдань комп'ютерного зору, що робить глибоке навчання домінуючим підходом машинного навчання в цій області. Розпізнавання та реідентифікація об'єктів, як одна з найпоширеніших задач комп'ютерного зору, широко вивчалася десятиліттями.

Згорткова нейронна мережа – алгоритм глибокого навчання, який може приймати вхідне зображення, присвоювати важливість (навчальні ваги та зміщення) різним об'єктам на зображенні та має можливість відрізняти один від іншого. Попередня обробка, необхідна для роботи з цим типом мереж, значно нижча порівняно з іншими алгоритмами класифікації, тоді як ранні дослідження базуються на побудові неглибоких моделей з низькорівневими ознаками об'єкту, створеними вручну [1].

Реідентифікація об'єктів зазвичай включає три етапи: виявлення об'єкту, вилучення особливостей та класифікація. Перший етап виконується окремо як шаг попередньої обробки зображення за допомогою моделей детекції, але на практиці не завжди виникає потреба у етапі виявлення [2]. У даному розділі будуть детально розглянуті етапи вилучення ознак та класифікації.

1.1 Класифікаційна згорткова нейронна мережа

Задачу реідентифікації об'єктів можна звести до задачі класифікації, але з динамічною кількістю класів, тобто з часом кількість класів може змінюватися. Розглянемо архітектуру згорткової нейронної мережі для класифікації зображень та принципи її роботи.

Згорткова мережа складається з різних видів шарів: згорткових шарів, агрегувальних (субдискретизацуючих) шарів та повноз’єднаних шарів, як показано на рисунку 1.1.

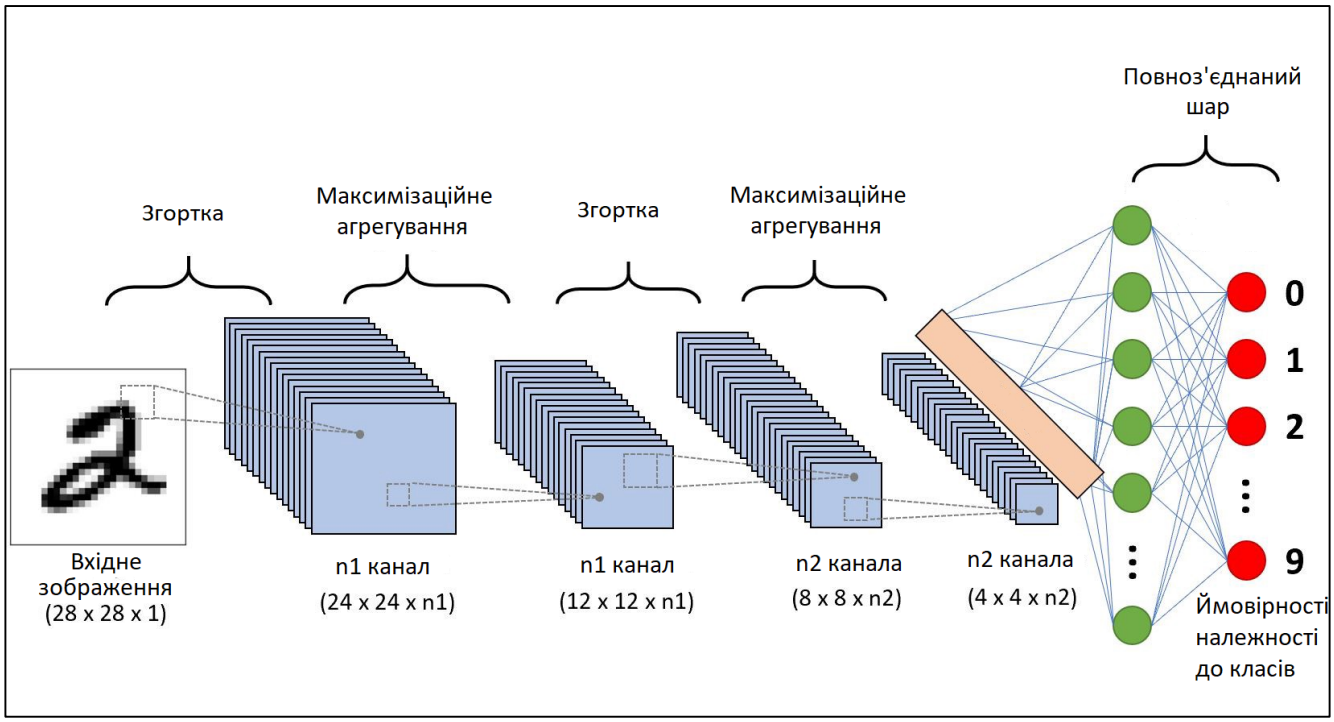

Рисунок 1.1 – Загальна архітектура згорткових нейронних мереж

Шари згортки та агрегування чергуються один за одним, формуючи вхідний вектор для повноз’єднаного шару.

Згортковий шар складається з набору ядер для згортки, де ядро – деякий тензор, значення якого спочатку ініціалізується випадково, а у ході навчання змінюється для покращення результату. Кожне ядро рухається по вхідному тензору як ковзне вікно, виконуючи операцію згортки згідно с формулою 1.1.

$$(f * g)[m,n] = \sum_{\substack{0<i<k \\ 0<j<l}} f[m-i,n-j] * g[i,j] \tag{1.1}$$

де f – вхідний тензор,

g – ядро згортки,

m, n – ширина та висота вхідного тензора,

k, *l* – ширина та висота ядра.

На виході згорткового шару отримуємо новий тензор, глибина якого дорівнює кількості ядер згортки, висота та ширина залишаються незмінними. Такий підхід дозволяє сильно зменшити кількість параметрів мережі, відносно повнозв'язних мереж. Також, операція згортки легко розпаралелюється на декілька ядер процесора.

Шари агрегування зменшують розмірність вхідного тензора попереднього шару. Це дозволяє прискорити обчислення та виконує роль регуляризації. Існує багато способів агрегації, наприклад, максимізаційне агрегування. Суть полягає у тому, що ми послідовно проходимо ділянки 2x2 вхідного тензора, залишаючи лише максимальне значення активації, тож висота та ширина тензора зменшуються в 2 рази.

Також, необхідно додати нелінійну функцію активації, щоб мережа могла вивчити складні патерни на зображені, що неможливо застосовуючи лише лінійні функції. Зазвичай використовується функція ReLU, що обраховується за формулою 1.2.

$$f(x) = x^+ = \max(0, x) \tag{1.2}$$

де *x* – вхідний тензор.

Завдяки своїй простоті, ReLU дозволяє швидше та ефективніше навчати згорткові нейронні мережі, ніж інші функції активації, наприклад, сигмоїда. Навіть такої простої нелінійності достатньо аби мережа вивчила патерни будь-якої складності, бо велика глибина нейронної мережі дозволяє останнім шарам вивчати дуже складні ознаки, базуючись на низькорівневих, що приходять з попередніх шарів [3].

Останній, повноз’єднаний, шар виконує роль лінійного класифікатора ознак, що надходять з останнього шару згортки або агрегації. Кількість нейронів у ньому дорівнює кількості класів у виборці. Вихід шару розраховується за формулою 1.3.

$$f(x) = (xW^T) + b \tag{1.3}$$

де x – вхідний вектор,

W – матриця вагових коефіцієнтів,

b – вектор коефіцієнтів зсуву.

Щоб отримати розподілення ймовірностей класів, до яких належить зображення, застосовується функція Softmax, що розраховується за формулою 1.4.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{1.4}$$

де z – вхідний вектор,

K – довжина вектору,

i – індекс у вихідному векторі.

Сума вихідного вектору після Softmax дорівнює 1, тож ці значення можна використовувати як ймовірності належності до певного класу об’єкту на зображенні та обрати клас з максимальною ймовірністю, як передбачений мережею.

Для навчання необхідно ввести функцію похибки, значення якої буде мінімізуватися за допомогою методу зворотного поширення помилки, що є ітеративним градієнтним алгоритмом. Класичною функцією похибки для багатокласової класифікації є перехресна ентропія, що розраховується за формулою 1.5.

$$Loss(y,p) = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \quad (1.5)$$

де M – кількість класів,

y – бінарний індикатор: 1, якщо клас c коректний для зображення o, інакше 0,

p – передбачена ймовірність класу c для зображення o.

Така архітектура дозволяє ефективно вирішувати завдання класифікації зображень з великою точністю, але для вирішення задачі реідентифікації повстають певні труднощі.

Динамічна зміна кількості класів. Навіть якщо у рамках задачі нам не потрібно відрізняти між собою небачені раніше класи об’єктів або об’єкти, додавання класу до такої мережі є дуже складною операцією. Розмір матриці останнього повнозв’язного шару залежить від кількості класів. Якщо змінюється кількість класів, нам потрібно змінити і матрицю. Доповнити її стовпчиком для нового класу, а потім довчити лише на зразках нового класу не вийде, бо це важлива архітектурна зміна. Тож, зміна кількості класів веде до повного навчання згорткової мережі з нуля, змінивши розмір вихідного шару.

Мала вибірка. Зазвичай, у задачах реідентифікації ми маємо малу кількість зразків до кожного класу, наприклад, для задачі розпізнавання співробітників на вході у офіс ми будемо мати, у кращому випадку, декілька зображень кожного співробітника. Але сучасні згорткові нейронні мережі потребують набагато більше зразків у вибірці для досягнення достатньої точності та узагальнюючої здатності.

Лінійна роздільність. У згортковій нейронній мережі класифікатором виступає лінійний повнозв’язний шар, тож він задає гіперплощини, що розділяють простір ознак, виділених згортковою частиною мережі. Оптимізуючи функцію похибки класифікації, простір ознак і сама гіперплощина змінюються для досягнення максимальної лінійної роздільності.

Лінійна роздільність – це властивість двох множин векторів в багатовимірному просторі: дві множини є лінійно роздільними, якщо існує хоча б

одна гіперплощина у просторі, така, що всі крапки однієї множини розташовані з одного боку гіперплощини, а іншої – з іншого (див. рис. 1.2).

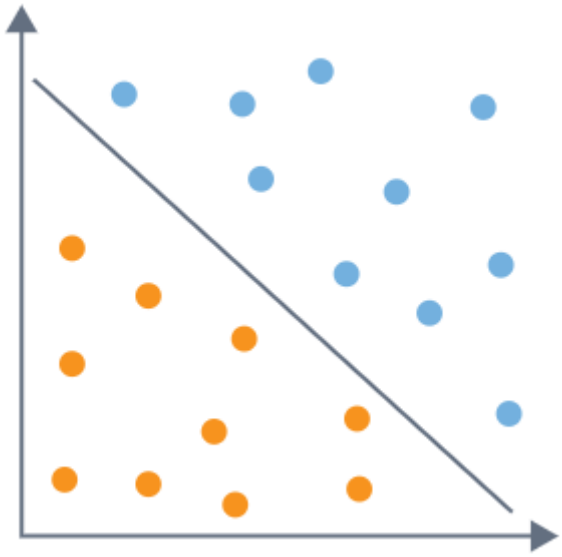

Рисунок 1.2 – Приклад двомірної лінійної роздільності

Тож, зразки близько до гіперплощини за наявності навіть малого шуму можуть бути розпізнані невірно. Також, через це ми не маємо змоги дізнатися, що тестовий зразок належить новому, небаченому раніше класу.

1.2 Методи навчання ознак

Очевидно, що класична згорткова нейронна мережа для класифікації має досить вагомі недоліки, тож для вирішення задач реідентифікації та розпізнавання застосовують інший підхід, заснований на згорткових нейронних мережах – метод навчання ознак.

Метод навчання ознак у комп'ютерному зорі полягає у пошуку відповідного простору ознак, у якому подібності між парами зображень зберігають відповідну структуру відстані, тобто дистанція між зображеннями одного класу або одно об’єкту значно менша, ніж дистанція між зображеннями різних класів. Такий

простір ознак також може покращити ефективність пошуку зображень, зокрема коли кількість категорій чи об'єктів дуже велика або невідома. Класичні метричні методи навчання вивчали випадок знаходження кращої відстані махаланобіса в лінійному просторі [4].

Сутність полягає у переході від задачі класифікації до задачі регресії, тобто замість оптимізації метрик, пов’язаних з розподілом ймовірностей цільових класів, оптимізуються певні метрики відстані між векторами ознак зображень.

Цей напрямок є достатньо новим, відносно класичних задач комп’ютерного зору та стрімко розвивається останні роки, бо з’являється усе більше великих датасетів з різних доменів. Архітектури таких моделей базуються на розглянутих згорткових нейронних мережах, але за згортковою частиною мережі слідує не повнозв’язний шар з Softmax, а інші конструкції для вирішення задачі регресії.

Навчання ознак вирішує перераховані у попередньому розділі проблеми згорткових мереж класифікації:

- кількість класів у тестовій виборці не грає великої ролі, бо кожний клас утворює свій «кластер» у просторі ознак, тож отримуючи нове зображення можна зробити висновок чи належить цей зразок до якогось з існуючих класів або це новий, небачений раніше, клас. Для досягнення такого ефекту потрібен великий датасет з відповідного домену, що може і не містити схожих на тестові зображень, але, завдяки великій кількості зразків, вектори ознак на виході згорткової мережі матимуть достатню узагальнюючу здатність;
- проблема малого датасету для конкретної задачі вирішується аналогічно описаному вище: достатньо мати великий датасет відповідного домену, що може мати інші цільові класи, тож з деякої точки зори методи навчання можуть поєднувати методи навчання як з учителем, так и без нього;
- методи навчання ознак не вирішують задачу лінійної роздільності векторів ознак після проходження через згорткову мережу, а напряму зменшують відстань між зразками одного класу та збільшують відстань між різними, тож класи стають відособленими у просторі ознак

(див. рис. 1.3), що дозволяє як класифікувати відомі класи, так і знаходити у тестовій вибірці нові класи.

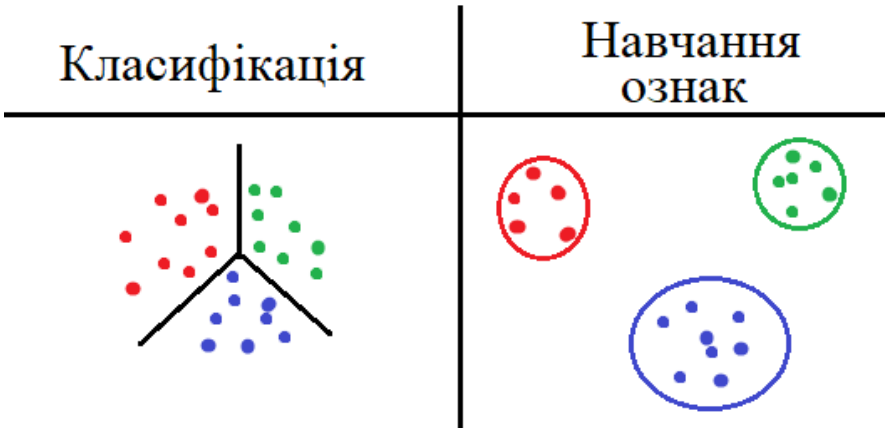

Рисунок 1.3 – Схематичне зображення просторів ознак

Тож, методи навчання ознак якнайкраще підходять для вирішення задачі реідентифікації об’єктів на зображеннях.

1.2.1 Сіамська нейронна мережа

Одним с перших методів навчання ознак є сіамська нейронна мережа. Сутність роботи полягає у тому, що зразки надходять до мережі парами, з кожного зразка виділяють вектор ознак за допомогою глибокої нейронної мережі (зазвичай, згорткової), а потім обчислюється відстань між ними за формулою евклідової відстані:

$$f(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1.6}$$

де p, q – отримані вектори ознак,

n – розмір вектору.

Отриману відстань можна порівнювати з деяким порогом, підібраним на валідаційній вибірці, для визначення належать обидва зразки до одного класу/об’єкту чи ні.

Архітектуру сіамської нейронної мережі зображено на рисунку 1.4. Згорткова частина має таку саму структуру, як згорткові нейронні мережі для класифікації зображень, але відсутній класифікаційний шар з Softmax. Також слід зазначити, що ваги згорткових частин для обох зразків однакові, тобто вони потрапляють у один і той самий простір ознак [5].

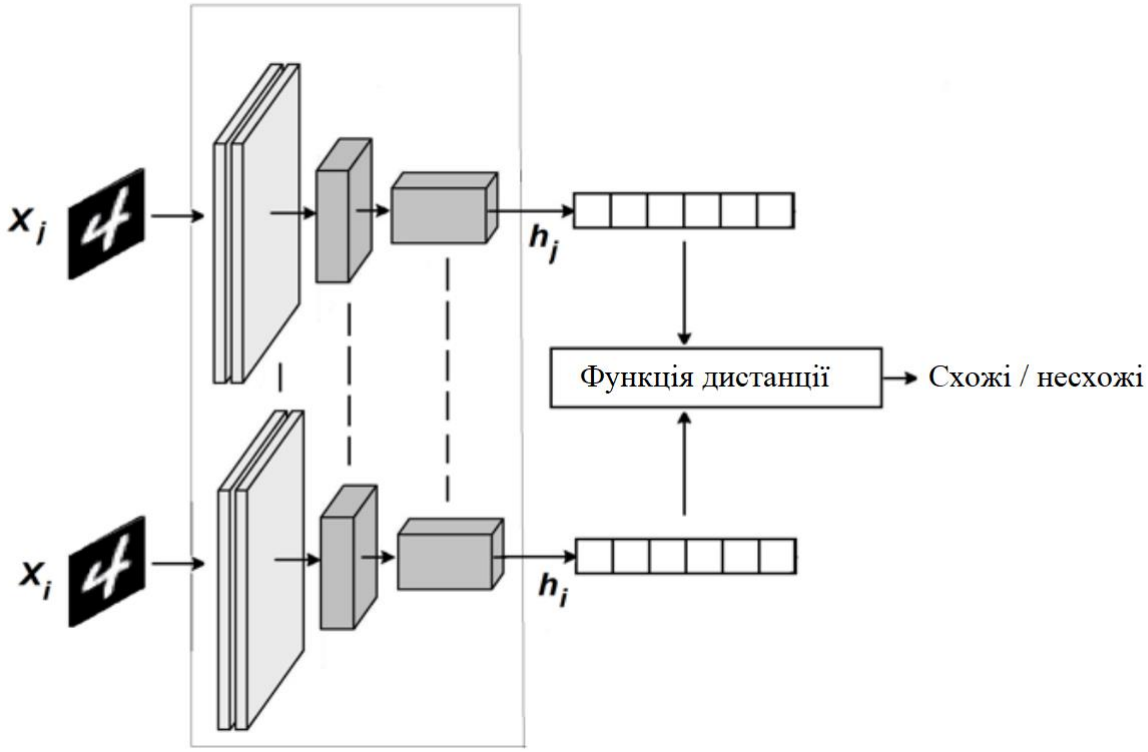

Рисунок 1.4 – Архітектура сіамської нейронної мережі

Для навчання такої мережі потрібна інша функція похибки, що буде мінімізувати відстань між схожими об’єктами та максимізувати відстань між різними. Так як ваги згорткової частини однакові для обох зображень, то під час навчання нейронної мережі методом зворотного поширення помилки, градієнти обох наборів згорткових ядер усереднюють и пропускають через одні ваги згорткової мережі.

У сучасних підходах для навчання сіамських мереж використовується контрастна функція похибки:

$$l(h_i, h_j, z_{ij}) = (1 - z_{ij})\|h_i - h_j\|_2^2 + z_{ij}\, max\left(0, \tau - \|h_i - h_j\|_2^2\right) \qquad (1.7)$$

де h_i – вектор ознак першого зображення,

h_j – вектор ознак другого зображення,

z_{ij} – бінарний показник, що відображає належність або неналежність обох зображень до одного класу, $z_{ij} \in \{0, 1\}$,

τ – попередньо визначений поріг.

Для вирішення задачі реідентифікації за допомогою сіамської нейронної мережі достатньо зберігати отримані вектори відомих нам зразків у базі даних або у будь-якій зручній структурі даних, наприклад у k-d дереві. Тоді, отримавши тестовий зразок, вилучимо з нього вектор ознак і порівняємо по попередньо визначеному порогу з найближчими векторами ознак відомих зображень. Якщо відстань буде меншою за поріг, вважаємо що це той самий клас об’єктів чи об’єкт, а якщо відстань перевищує – на вхід було подано зразок нового класу.

Цікаво порівняти отриманий простір ознак з простором ознак звичайної класифікаційної згорткової нейронної мережі, але зазвичай вектор ознак має сотні вимірів, що ускладнює можливість перевірити гіпотезу про лінійну роздільність класів у отриманому згортковою мережею просторі.

Для візуалізації можна використати методи зменшення розмірності, наприклад, метод головних компонент. Сутність методу полягає у великій кількості послідовних ортогональних перетворень з мінімізацією втрати інформації, але на практиці цей метод не підходить для візуалізації векторів ознак нейронних мереж.

Тому використаємо більш сучасний метод – *t*-розподілене вкладення стохастичної близькості, що був розроблений Лоренсом ван дер Маатеном і Джефрі Хінтоном у 2008 році. Головні відмінності цього методу від методу головних компонент:

– *t*-розподілене вкладення стохастичної близькості є градієнтним методом, тобто навчається подібно до нейронної мережі;
– метод головних компонент використовує лінійні матричні перетворення, тоді як другий метод – нелінійні;
– *t*-розподілене вкладення стохастичної близькості вирішує задачу пропорційного збереження евклідової відстані у вихідному та цільовому просторах, тоді як перший метод видаляє виміри, у яких дисперсія ознак найменша.

Останній пункт є найбільшим плюсом для візуалізації простору ознак, бо ми досліджуємо наскільки вектори ознак класів лінійно роздільні у отриманому просторі.

Для перевірки навчимо класифікаційну згорткову нейрону мережу на датасеті зразків рукописного написання цифр MNIST [6] та відобразимо вектори ознак з останнього згорткового шару у двовимірний простір за допомогою *t*-розподіленого вкладення стохастичної близькості, результат зображено на рисунку 1.5.

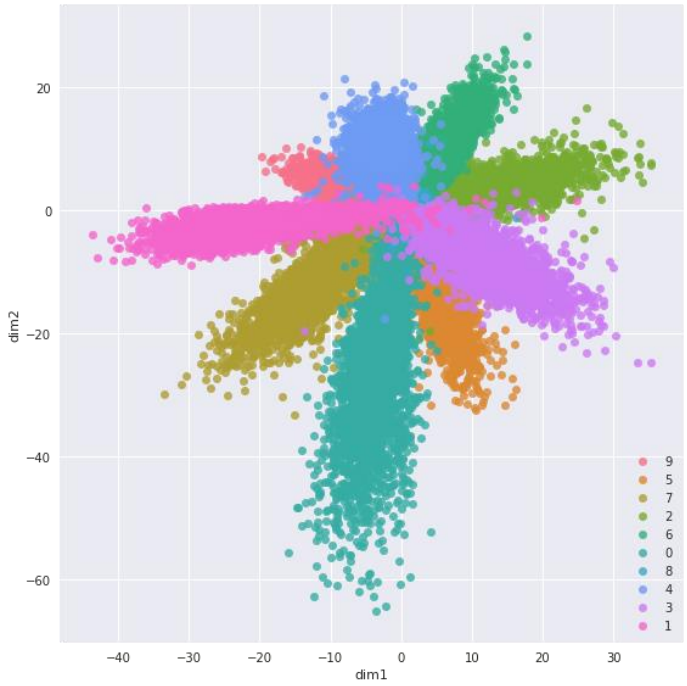

Рисунок 1.5 – Двомірне відображення простору ознак згорткової мережі

Бачимо, що відстань між класами майже відсутня, зразки майже зливають в одне ціле, однак є можливість провести лінії для розділення простору по класам, тобто отриманій простір є лінійно розподільним.

Тепер проведемо той самий експеримент з глибокою згортковою сіамською нейронною мережею (див. рис. 1.6).

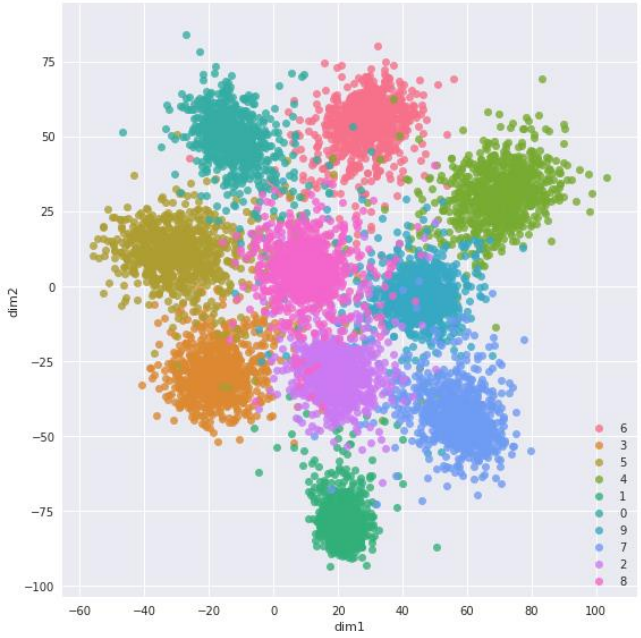

Рисунок 1.6 – Двомірне відображення простору ознак сіамської мережі

Можна побачити, що зразки кожного класу сформували кластери, мінімізуючи відстань усередині класу та максимізуючи відстань між зразками різних класів. Такий простір ознак найкраще підходить для задач реідентифікації та розпізнавання об’єктів.

Тож, сіамські нейронні мережі можуть використовуватися для ефективного вирішення задачі реідентифікації зображень та вирішують основні проблеми класичних згорткових мереж у рамках цієї задачі.

Але ця модель досить наївна, наприклад, нам не завжди потрібно зближувати зразки, що вже знаходяться у центрі кластеру свого класу у просторі ознак. Також, немає гарантії, що евклідова відстань є оптимальною мірою дистанції для побудови простору ознак.

1.2.2 Нейронна мережа SphereFace

SphereFace – це метод навчання ознак, що базується на згорткових нейронних мережах та модифікує останній лінійний шар класифікації з Softmax. Метод був розроблений для розпізнавання обличь, але його можна застосовувати у задачах реідентифікації будь-яких об’єктів. Головна ідея така сама, як і у сіамських мереж: перехід від задачі класифікації до задачі регресії, але відстань вимірюється між центром класу та зразком, а не між двома зразками. Ключовою особливістю є спосіб переводу зображення у простір ознак. Замість обчислення евклідової дистанції у прямокутному просторі, SphereFace переводить зображення у точки на гіперсфері, обчислюючи косинусну дистанцію між ними, як показано у формулі 1.8 [7].

$$cos(\theta) = \frac{A * B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n}(A_i * B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} * \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{1.8}$$

де A, B – вектори однакової розмірності,

n – довжина векторів,

θ – кут між векторами.

Для досягнення такого ефекту достатньо змінити архітектуру останнього шару згорткової мережі та модифікувати функцію похибки. Представимо формулу функції помилки Softmax з урахуванням ваг та зсуву останнього лінійного класифікаційного шару:

$$L = -log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_j e^{W_{y_j}^T x_i + b_{y_j}}}\right) \tag{1.9}$$

де W – матриця ваг лінійного шару,

b – зсув лінійного шару,

x – вектор ознак,

y – цільовий клас.

Представимо добуток вектору x та матриці W у вигляді добутку їх норм на косинус куту між ними виходячи з формули 1.8:

$$L = -log\left(\frac{e^{\left\|W_{y_i}\right\|\|x_i\|\cos\left(\theta_{y_i,i}\right)+b_{y_i}}}{\sum_j e^{\left\|W_{y_j}\right\|\|x_i\|\cos\left(\theta_{y_j,i}\right)+b_{y_j}}}\right) \quad (1.10)$$

де W – матриця ваг лінійного шару,

b – зсув лінійного шару,

x – вектор ознак,

y – цільовий клас,

θ – кут між векторами ($0 \leq \theta_{j,i} \leq \pi$).

За умови нормалізації матриці W для отримання норми матриці рівної 1 та занулення зсуву у формулі 1.10 отримуємо модифіковану функцію втрат Softmax, що у процесі навчання методом зворотного поширенням помилки оптимізує косинусу відстань між вектором ознак та центром ознак відповідного класу:

$$L = -log\left(\frac{e^{\|x_i\|\cos\left(\theta_{y_i,i}\right)}}{\sum_j e^{\|x_i\|\cos\left(\theta_{y_j,i}\right)}}\right) \quad (1.11)$$

де x – вектор ознак,

y – цільовий клас,

θ – кут між векторами ($0 \leq \theta_{j,i} \leq \pi$).

Отримана функція значно простіша для обчислення, що прискорює процес навчання мережі, також вона враховує косинусні міри на гіперсфері, але це не обов'язково приводить до вивчення дискримінаційних ознак, тож автори SphereFace додали до модифікованої версії Softmax мультиплікативне кутове розділення і отримали A-Softmax:

$$L = -log\left(\frac{e^{\|x_i\|\cos\left(m\theta_{y_i,i}\right)}}{e^{\|x_i\|\cos\left(m\theta_{y_i,i}\right)} + \sum_{j\neq y_i} e^{\|x_i\|\cos\left(\theta_{y_j,i}\right)}}\right) \qquad (1.12)$$

де x – вектор ознак,

y – цільовий клас,

θ – кут між векторами,

m – мультиплікативний коефіцієнт кутового розділення.

Завдяки коефіцієнту m досягається кутова роздільність центрів класів на гіперсфері. Слід зазначити, що при $m=1$ формула приймає вид модифікованої функції втрат Softmax.

Коефіцієнт m створює уявну границю кожного класу на гіперсфері навколо кожного центрового вектору. Головна ідея цього у тому, що модель не отримує штраф за те, що отриманий вектор ознак не співпадає з центром цього класу, при умові що той лежить у границях, заданих коефіцієнтом.

Таким чином, коефіцієнт m дозволяє регулювати допустиму похибку нейронної мережі при реідентифікації об'єкту. На перший погляд це лише маленька деталь, але на практиці під час навчання центри векторів стабілізуються та стають незмінними значно швидше, ніж без цього параметру.

Геометричну інтерпретацію розглянутих функцій втрат зображено на рисунку 1.7, на якому W – центральний вектор деякого класу, x – вхідний вектор, O – центр гіперсфери, θ – кут між вхідним вектором та центральним вектором деякого класу.

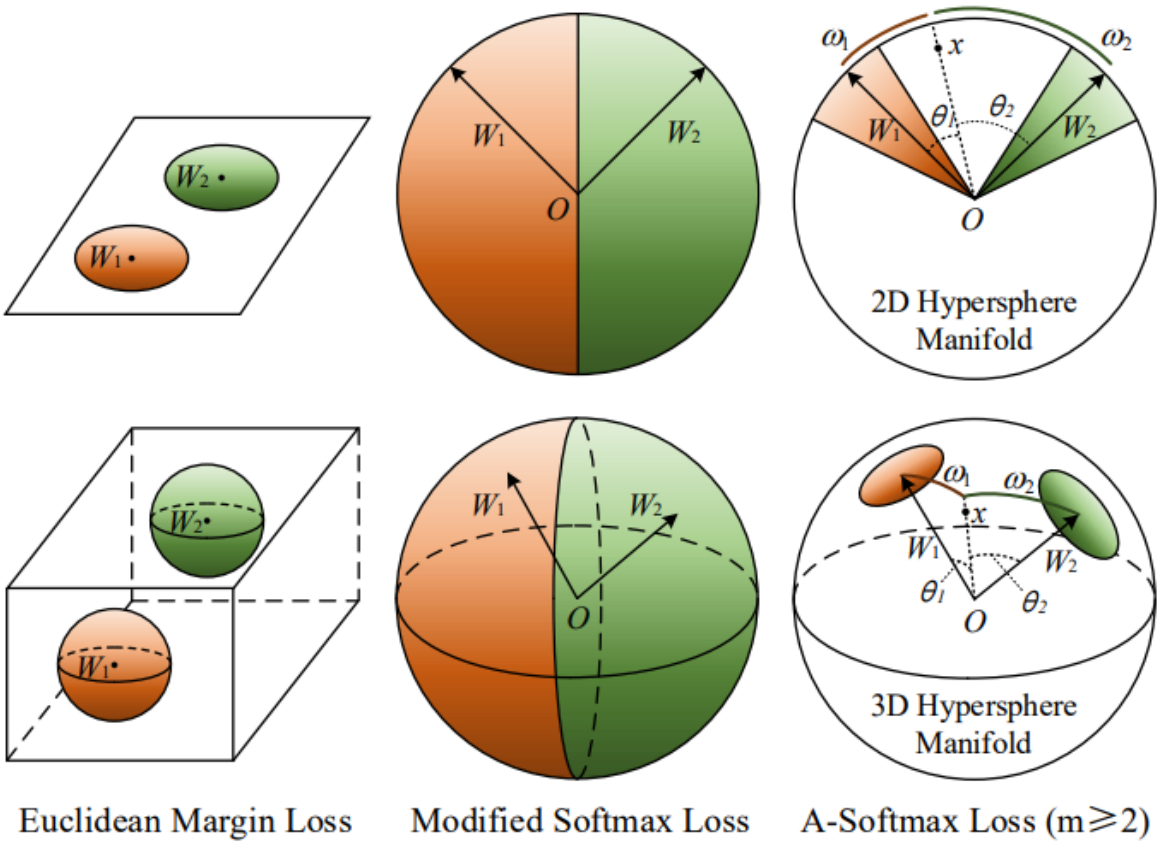

Рисунок 1.7 – Геометрична інтерпретація функцій помилки

Для перевірки кутової роздільності, візуалізуємо ознаки зразків з датасету рукописних цифр у двомірному просторі (див. рис. 1.8). Очевидно, що при збільшенні m вивчені ознаки стають значно більш дискримінаційними через більший міжкласову кутову роздільність.

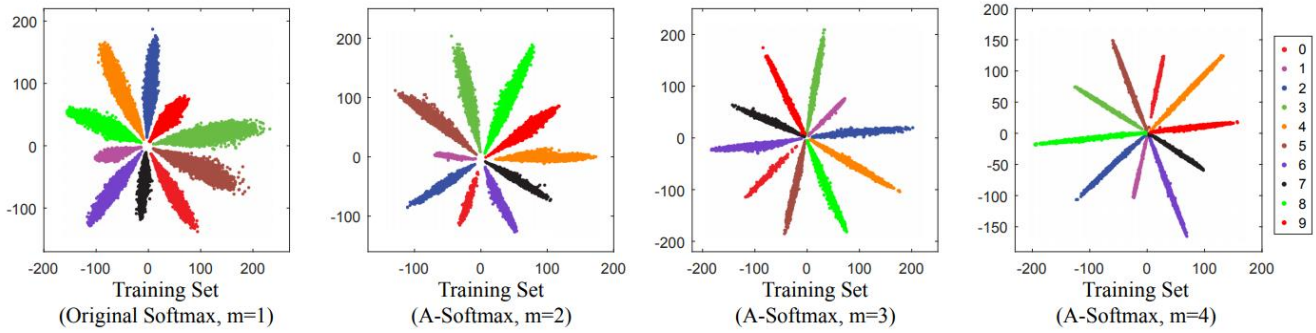

Рисунок 1.8 – Двомірне відображення простору ознак мережі SphereFace

Такі особливості роблять A-Softmax ефективним методом навчання ознак, що дозволяє використовувати його у задачах розпізнавання та реідентифікації.

1.2.3 Нейронна мережа ArcFace

ArcFace – це остання модифікація SphereFace, що досягає найвищих показників метрик якості на більшості датасетів з розпізнавання об’єктів та має наступну формулу [8]:

$$L = -log\left(\frac{e^{\|x_i\|\cos\left(\theta_{y_i,i}+m\right)}}{e^{\|x_i\|\cos\left(\theta_{y_i,i}+m\right)}+\sum_{j\neq y_i}e^{\|x_i\|\cos\left(\theta_{y_j,i}\right)}}\right) \quad (1.13)$$

де x – вектор ознак,

y – цільовий клас,

θ – кут між векторами,

m – адитивний коефіцієнт кутового розділення.

Єдина відмінність ArcFace від SphereFace – кутова роздільність досягається завдяки адитивному коефіцієнту, замість мультиплікативного. Автори мотивують це отриманою формою роздільності у гіперпросторі ознак (див. рис. 1.9), о на їхній погляд найкраще підходить для задач класифікації та реідентифікації об’єктів на зображеннях.

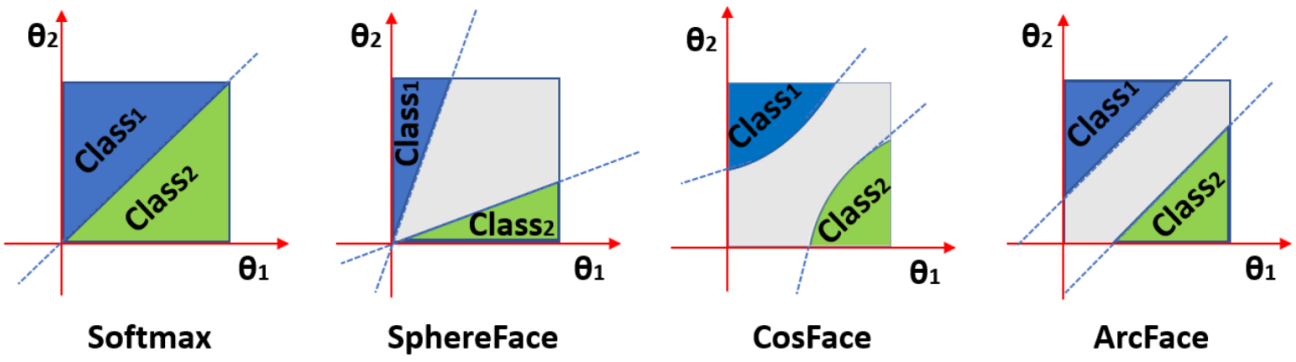

Рисунок 1.9 – Межі прийняття рішень різних функцій втрат

Тож, на даний момент ArcFace є найбільш перспективним методом навчання ознак з простою архітектурою та інтуїтивною геометричною інтерпретацією, але головним недоліком є нестабільний процес навчання.

1.3 Постановка задачі

Реідентифікація об’єктів – це завдання асоціювання зображень одного і того ж об’єкта чи класу об’єктів, знятих з різних камер або з однієї і тієї ж камери в різних випадках; складається з двох етапів: вилучення ознак та класифікація.

Для вилучення ознак використаємо модель ArcFace, що переводить зображення у вектор у просторі ознак. Простір являє собою гіперсферу, на якій зразки є точками. Також, у ході навчання ArcFace формуються центри класів.

Для класифікації вектору ознак можна порівнювати косинусну відстань від нього до центрів усіх класів з деяким порогом, що підбирається на валідаційній вибірці. Тобто, якщо відстань до деякого центру менша за поріг, то відповіддю буде відповідний клас, інакше вважаємо, що отримане зображення відноситься до раніше небаченого класу чи об’єкту.

Формально можемо виділити три основні задачі реідентифікації: реідентифікація з відкритим набором даних, реідентифікація з закритим набором даних та попарна реідентифікація. Усі ці задачі модель ArcFace може вирішувати без архітектурних змін.

1.3.1 Реідентифікація з відкритим набором даних

Реідентифікація з відкритим набором даних передбачає, що під час тестування ми можемо отримувати на обробку зразки класу, що був відсутній у

навчальному датасеті. Практичним прикладом є камера спостереження у службовому закладі: система має базу даних співробітників та може їх реідентифікувати, але також у поле зору можуть потрапити люди, що були відсутні у тренувальному датасеті та базі даних.

Для тренування та валідації візьмемо датасет MegaFace [9]. MegaFace – це широкомасштабний набір даних з розпізнавання облич. До датасету входить 4753330 обличчя, що належать 672057 особам з 3311471 фотографій, завантажених з 48383 фотоальбомів користувачів Flickr. Усі фотографії включали ліцензію Creative Commons, але більшість не була ліцензована для комерційного використання. Приклад зображень з датасету можна побачити на рисунку 1.10.

Рисунок 1.10 – Зображення з датасету MegaFace

Для валідації задачі з невідомою кількістю класів має сенс використати точність як метрику якості мережі. Для формалізації задачі для початку

сформулюємо метрики точності для задачі зведеної до бінарної, тобто розглядаємо кожен клас окремо:

– *TP* – true positive – правильно ідентифікований клас;

– *TN* – true negative – правильно відхилений клас;

– *FP* – false positive – помилка першого роду – помилкове спрацювання;

– *FN* – false negative – помилка другого роду – пропуск класу.

Тоді задамо метрику точності наступною формулою:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.14}$$

де *acc* – точність.

Тож, для валідації поточної задачі достатньо буде рахувати точність по кожному класу, а потім порахувати середнє значення, приймаючи до уваги, що невідомі фотографії віднесено до окремого класу, що був відсутній у навчальній вибірці.

1.3.2 Реідентифікація з закритим набором даних

Під час реідентифікація з закритим набором даних на етапі тренування ми маємо зразки усіх класів, що будуть застосовані для валідації, тобто задача зводиться до звичайної класифікації зображень.

Для цієї задачі візьмемо датасет CIFAR-10 [10]. Набір даних CIFAR-10 складається з 60000 кольорових зображень 32x32 в 10 класах, по 6000 зображень на клас. Є 50000 навчальних зображень та 10000 тестових зображень. Цей датасет широко використовується для оцінки якості різноманітних моделей комп’ютерного зору для задач класифікація з закритим набором даних. Приклади зображень представлені на рисунку 1.11.

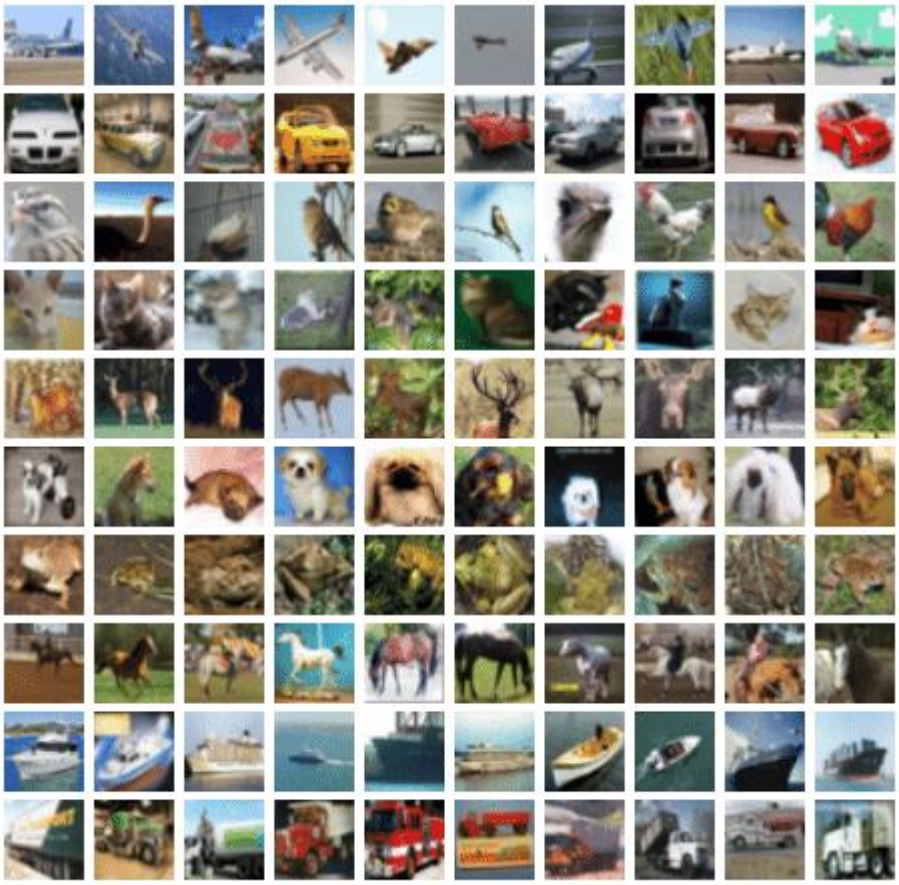

Рисунок 1.11 – Зображення з датасету CIFAR-10

Для валідації задачі класифікації використаємо метрику *FI*-micro, заданою наступною формулою:

$$precision = \frac{TP}{TP + FP} \tag{1.15}$$

$$recall = \frac{TP}{TP + FN} \tag{1.16}$$

$$f1 = \frac{2 * presicion * recall}{precision + recall} \tag{1.17}$$

де *TP*, *FP*, *FN* – див. попередній підпункт,

precision – позитивно прогнозована величина,

recall – чутливість,

f1 – метрика *F1*-micro.

Така метрика підходить для задач з незбалансованими класами [11], що має сенс у більшості задач реідентифікації, бо кількість зразків кожного класу може коливатися дуже сильно.

1.3.3 Попарна реідентифікація

У рамках задачі попарної реідентифікації валідаційний датасет містить пари зразків і розмітку у виді єдиного значення true або false, тобто задача полягає у тому, щоб модель відповідала на єдине запитання – чи один і той самий об’єкт/клас знаходиться на обидвох зображеннях. При цьому класи тренувального та валідаційного наборів не перетинаються.

Набір даних Trillion-Pairs [12] містить 1,58M зображень з Flickr саме з такою розміткою. Приклади можна побачити на рисунку 1.12.

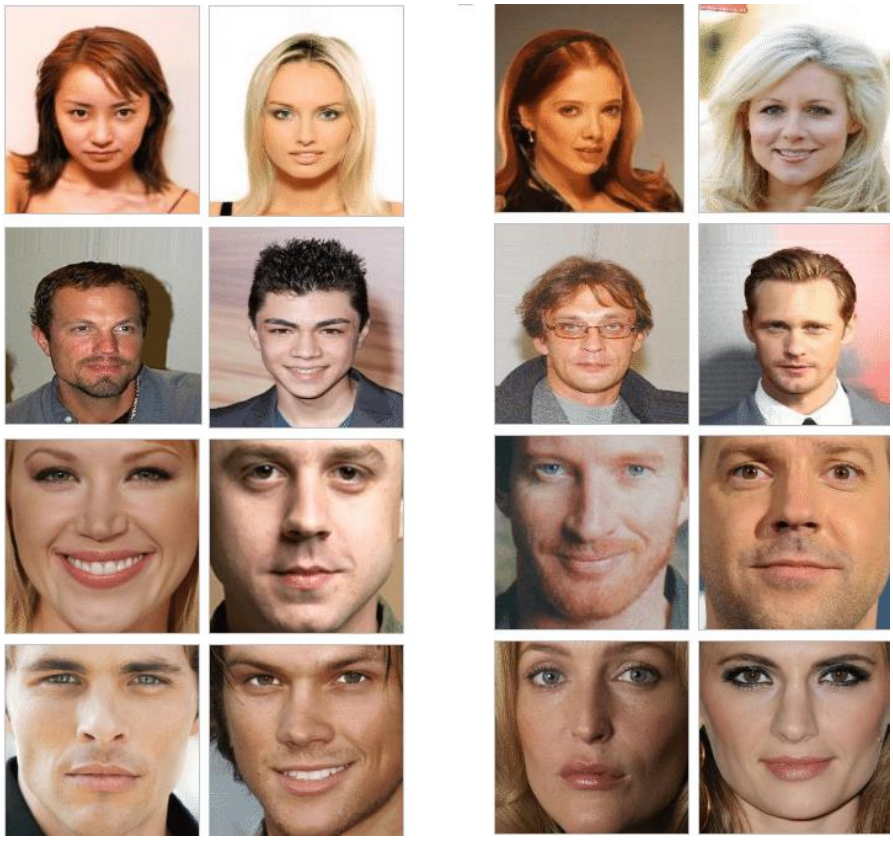

Рисунок 1.12 – Зображення з датасету Trillion-Pairs

Валідувати модель на такому датасеті можна і за точністю (див. формулу 1.14), але більшість алгоритмів перевіряють ефективність свого алгоритму на даному датасеті за допомогою метрики false positive rate, що рахується за формулою:

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{1.18}$$

де TN, FP – див. підпункт 1.3.1,

N – кількість зразків,

FPR – false positive rate.

Така метрика більш підходить до конкретної задачі, ніж точність, через те, що така задача специфічна для систем захисту за допомогою візуальної реідентифікації, наприклад, розблокування мобільного телефона. У розблокування телефона найголовніша задача запобігти хибне спрацювання, тобто запобігти потрапляння зловмисника у систему. Метрика FPR набагато ясніше відображає ймовірність таких ситуацій, ніж точність.

2 ОПИС ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

Метою модифікації ArcFace є стабілізація процесу навчання, бо головними проблемами цього методу є:

- розходження помилки під час навчання;
- перенавчання під тренувальні дані;
- велика кількість гіперпараметрів, які зазвичай відбираються вручну, наприклад, адитивний коефіцієнт кутового розділення.

Але крім процесу навчання також має сенс оптимізувати архітектуру мережі та інференс.

Інференс – процес виводу результату моделі, що у нашому випадку є процесом проходження зображення через нейронну мережу.

Авжеж, сучасні технології дозволяють шукати оптимальну архітектуру під датасет у автоматичному режимі, наприклад, технологія Neural architecture search [13] або алгоритм Біморф [14]. Але ці технології потребують дуже багато ресурсів, бо перебирають, навчають та валідують велику кількість варіантів архітектур на поточному датасеті. Також, зазвичай знайдена оптимальна архітектура виходить з великою кількістю шарів та інших операцій, що призводить до довгого інференсу.

Тож, у розділі розглянемо методи оптимізації усіх трьох складових: архітектури, навчання та інференсу.

2.1 Триплетний відбір зразків

Сутність алгоритму триплетного відбору зразків полягає у модифікації методу подачі зразків у нейронну мережу під час навчання ознак у сіамських мережах. Зазвичай, усі зразки випадково ділять на пакети та передають у мережу.

У цьому алгоритмі кожен пакет містить 3 зразки: якір, позитивний зразок, негативний зразок [15].

Якір – випадково вибраний зразок.

Позитивний зразок – зразок, що належить до того ж класу, що і якір, але знаходиться далеко від якорю.

Негативний зразок – зразок, що належить до іншого класу, але знаходиться дуже близько до якорю.

Тож, після навчання на такому пакеті зразків ми намагаємося виправити найскладніші випадки для розпізнавання мережею (див. рис. 2.1).

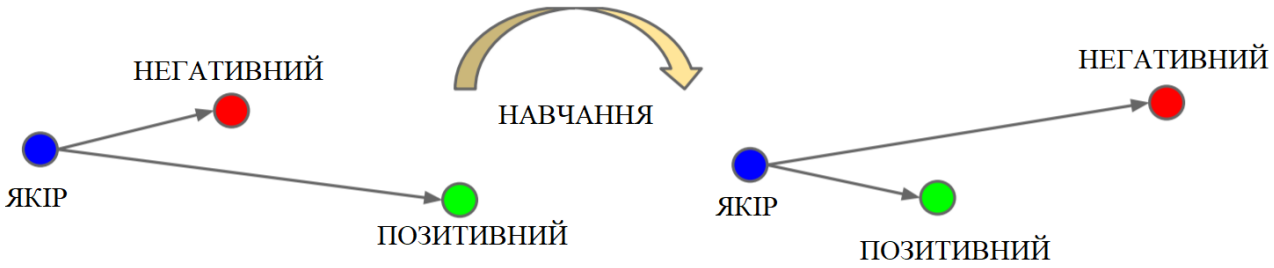

Рисунок 2.1 – Схематичне зображення класичного триплетного відбору зразків

Для використання такого методу з ArcFace достатньо випадково вибрати клас та вважати його якорем, тоді знайти найближчий зразок іншого класу та найбільш віддалений зразок цього класу за допомогою косинусної відстані. Такий підхід може прискорити та стабілізувати навчання.

Також, цей алгоритм має декілька стратегій:

– випадкова[16];
– всі позитивні, всі негативні [17];
– найпростіший позитивний, найтяжчий негативний [18].

Випадкова стратегія найпростіша: ми завжди обираємо довільний позитивний і довільний негативний зразки (див. рис. 2.2). Хоча така стратегія є досить наївною, але якість метрик сіамських нейронних мереж зазвичай підвищували за рахунок такої стратегії.

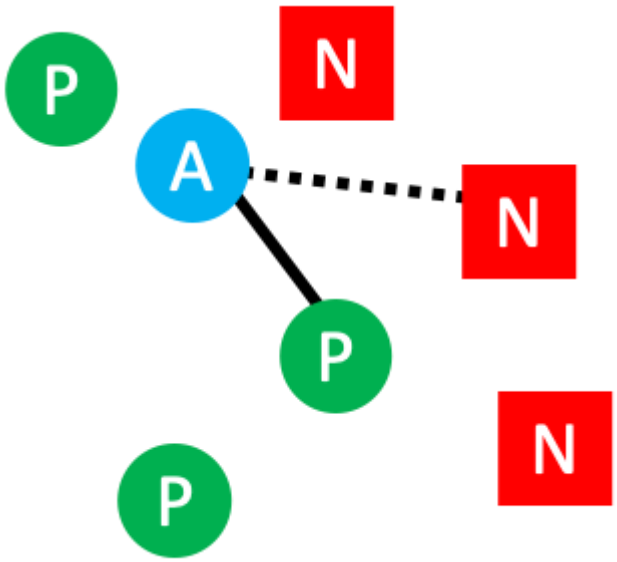

Рисунок 2.2 – Схематичне зображення випадкової стратегії триплетного відбору зразків

Стратегія «всі позитивні, всі негативні» є найбільш складною по ресурсам, необхідним для обробки: якір обирається випадково, а позитивними та негативними стають усі інші зразки (див. рис. 2.3).

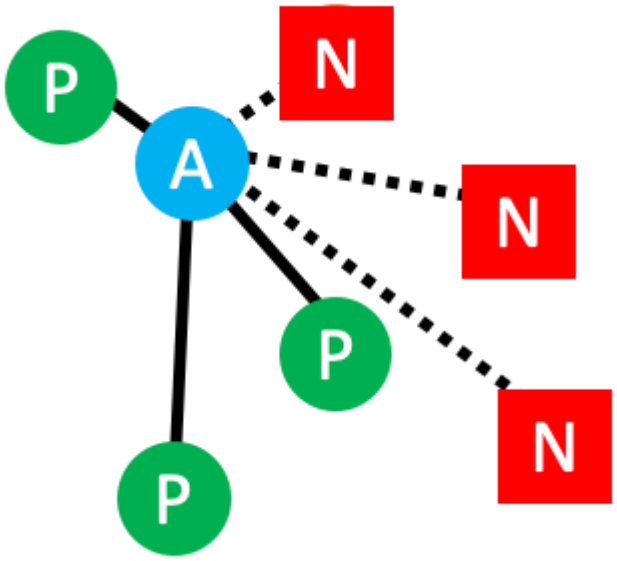

Рисунок 2.3 – Схематичне зображення стратегії «всі позитивні, всі негативні» триплетного відбору зразків

Стратегія «найпростіший позитивний, найтяжчий негативний» передбачає пошук найближчого до якорю зразка того ж класу та зразка іншого класу, що вимагає перебору поточних векторів ознак усіх зразків (див. рис. 2.4).

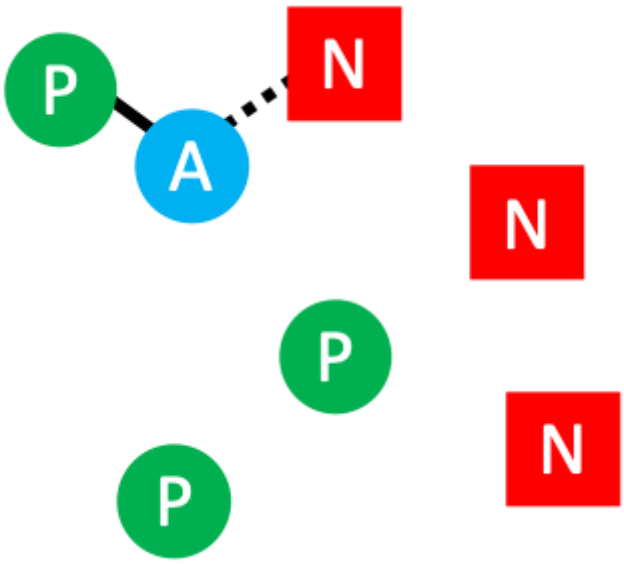

Рисунок 2.4 – Схематичне зображення стратегії «найпростіший позитивний, найтяжчий негативний» триплетного відбору зразків

Усі три стратегії є сучасними та актуальними, тож потрібно перевірити їхню ефективність на конкретній задачі та конкретній архітектурі нейронної мережі.

2.2 Нормалізація даних

Нормалізація пакетів даних – метод підвищення швидкості, продуктивності та стабільності навчання нейронних мереж. Метод використовується для нормалізації вхідного шару шляхом регулювання та масштабування активацій, що реалізується шляхом виконання функції для кожного пакету даних, описаної формулою [19]:

$$\widehat{x_i}^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{{\sigma_B^{(k)}}^2 + \epsilon}} \tag{2.1}$$

де x – вхідний вектор,

B – партія даних,

m – розмір партії,

σ – дисперсія виміру,

k – вимір,

ϵ – дуже мала константа для чисельної стійкості,

μ – середнє значення виміру.

Головний недолік цього методу: великі ризики при роботі з маленькими пакетами даних [20]. Очевидно, що нормалізація не може бути використана, коли розмір пакету даних дорівнює 1, але навіть трохи більші розміри партії можуть спричинити проблеми. В ідеалі ми хочемо використовувати глобальне середнє значення та дисперсію для нормалізації входів у шар. Однак, обрахування середнього значення для всього датасету після кожного оновлення мережі є занадто дорогим. Ось чому ми оцінюємо середнє значення та відхилення, використовуючи міні-пакетну статистику. Той факт, що середнє значення та дисперсія – це просто приблизні оцінки, означає, що вони містять певну кількість помилок і будуть змінюватися від пакету до пакету. Менші розміри міні-партії збільшують дисперсію цих оцінок, тобто ми повинні бути дуже обережними щодо розміру партії при використанні стохастичного градієнтного спуску з пакетною нормалізацією.

Тож, існують дещо інші алгоритми нормалізації: нормалізація ваг [21] та нормалізація шарів [22].

Нормалізація ваги – це метод, розроблений компанією Open AI, який замість нормалізації пакету даних, нормалізує ваги шару нейронної мережі. Цей процес можна описати наступною формулою:

$$w = \frac{g}{\|v\|} v \tag{2.2}$$

де w – ваги шару нейронної мережі,

g – скалярний параметр,

v – вектор параметр.

Параметри g та v знаходяться у процесі градієнтного спуску.

Завдяки цьому методу, ми відокремлюємо норму вагового вектору від його напрямку. Ця зміна динаміки полегшує оптимізацію та процес навчання. Крім того, середня величина і відхилення не залежать від партії даних, тож нормалізація ваги часто набагато швидша, ніж нормалізація партії.

Нормалізація шару дуже схожа на нормалізацію пакетів. Основна особливість нормалізації шару полягає в тому, що вона нормалізує вхідні дані вздовж ознак, а не вздовж екземплярів, як показано на рисунку 2.5.

Рисунок 2.5 – Схематичне зображення нормалізації шару та нормалізації пакетів

Додати шар з нормалізацією пакетів даних можна після останнього шару згортки. Це ускладнить архітектуру класифікатора мережі, що трохи сповільнить процес інференсу мережі, але може підняти як швидкість навчання, так і якість мережі.

2.3 Шар виключення

Однією з головних проблем глибоких мереж є перенавчання [23]. Проблема перенавчання полягає у тому, що після деякого етапу навчання модель починає лише запам’ятовувати тренувальні дані, втрачаючи узагальнюючу здатність на повному датасеті, що призводить до зниження якості роботи мережі на тестових даних (див. рис. 2.6).

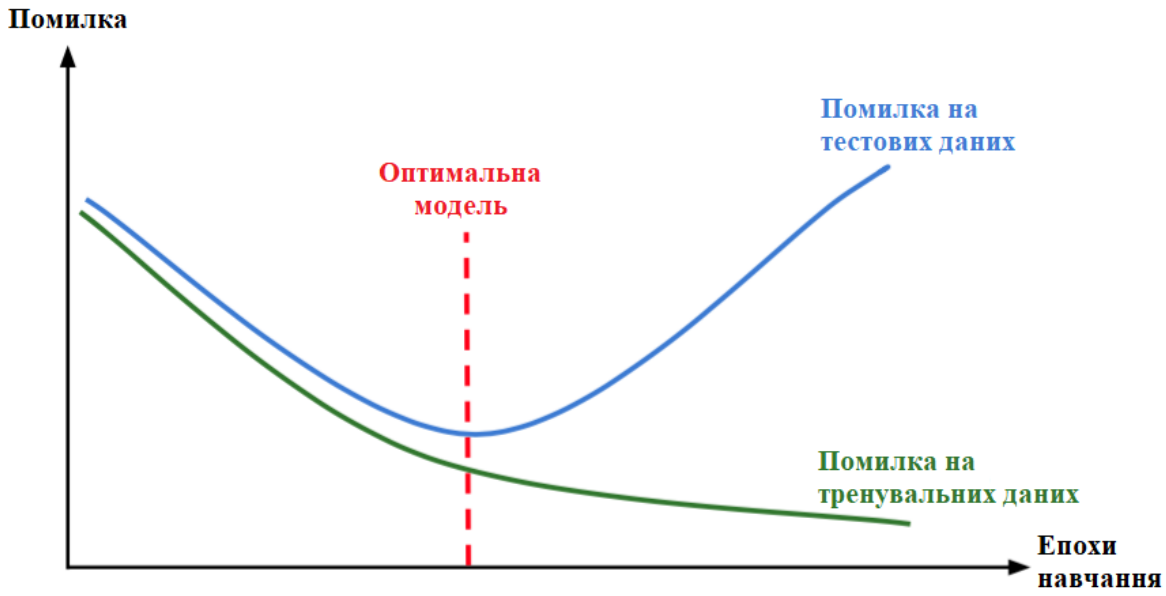

Рисунок 2.6 – Схематичне зображення перенавчання моделі машинного навчання

Для запобігання перенавчання застосовуються два основних підходи:

– рання зупинка навчання;

– регуляризація.

Рання зупинка навчання передбачає моніторинг якості моделі у процесі навчання з метою зупинити процес у разі зменшення метрик на тестових даних протягом деякої кількості кроків.

Шар виключення – це методика регуляризації для зменшення перенавчання в нейронних мережах за рахунок запобігання складних спільних адоптацій даних між собою. Метод успішно застосовуються в різних задачах, включаючи регуляризацію нейронної мережі, стиснення моделі та вимірювання невизначеності виходів нейронної мережі. Незважаючи на те, що спочатку метод було сформульовано для лінійних шарів нейронної мережі, останні досягнення зробили шар виключення також застосовним до згорткових та рекурентних шарів нейронної мережі [24].

Суть полягає у тому, що із повнозв’язного шару в процесі навчання виключаються певні нейрони, тобто їх активацію анулюють (див. рис. 2.7).

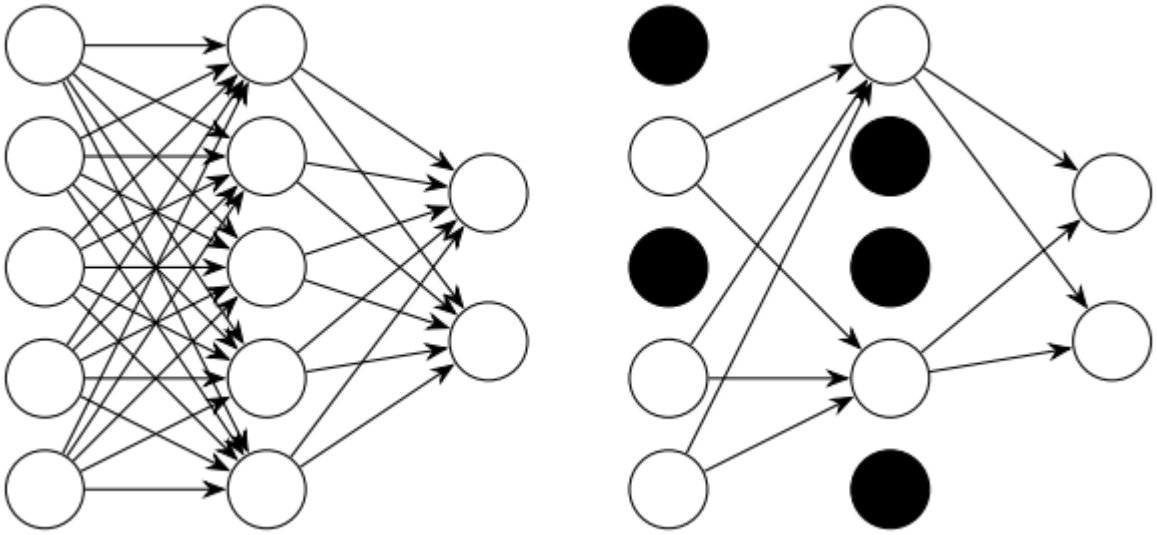

Рисунок 2.7 – Приклад роботи шару виключення

Завдяки цьому алгоритму можна зробити вектор ознак більш дискримінаційним, бо ознаки стануть більш незалежними одна від одної. Також, шар виключення прискорює навчання через зменшення вектору даних.

2.4 Аугментація зображень

Метою навчання є знаходження моделлю деяких узагальнюючих патернів поточного домену [25]. Однак, навчальна вибірка не містить усі можливі зразки домену, це не є можливим, її розмір обмежений. Чим більша вибірка – тим більшої узагальнюючої здатності можна досягти після навчання, але збір датасету та його розмітка дуже довге та складне завдання.

Штучно розширити вибірку можна за допомогою аугментації зображень [26]. Ідея дуже проста: деякі перетворення зображення (як афінні, так і не афінні) не змінюють сутність (правильну відповідь) для зображення. Наприклад, якщо горизонтально відобразити зображення людини, то це все ще залишиться зображенням людини, тож його не потрібно дорозмічувати, але для згорткової нейронної мережі це буде зовсім нове зображення, що несе корисну інформацію, бо фільтри такої мережі працюють локально та мають своє «поле зору». Приклади коректних аугментацій можна побачити на рисунку 2.8.

Рисунок 2.8 – Приклад роботи шару виключення

Авжеж, аугментації потрібно обирати обережно і відповідно до конкретного домену, у іншому разі якість розпізнавання мережею може значно знизитися. Наприклад, якщо перевернути зображення стільця на 180 градусів, то отримане зображення зовсім не збігається з можливими зображеннями під час тестування, тобто мережа буде навчатися хибним патернам. Таким чином, неприродньо підібрані аугментації створюють шум, що зазвичай знижує метрики якості, але деколи такий шум може виступати у ролі регуляризації, без перевірки на практиці під конкретну задачу та датасет сказати неможливо [27].

Також, аугментацію можна застосовувати не лише під час навчання. Існує спосіб аугментації, що призначений для покращення результатів на тестуванні: test time augmentation (TTA) – аугментація під час тестування. Ідея дуже проста: проведемо інференс на початковому зображенні, а потім на одному або кількох аугментованих зображеннях. Отримані вектори ознак агрегуємо, наприклад, як зображено на рисунку 2.9.

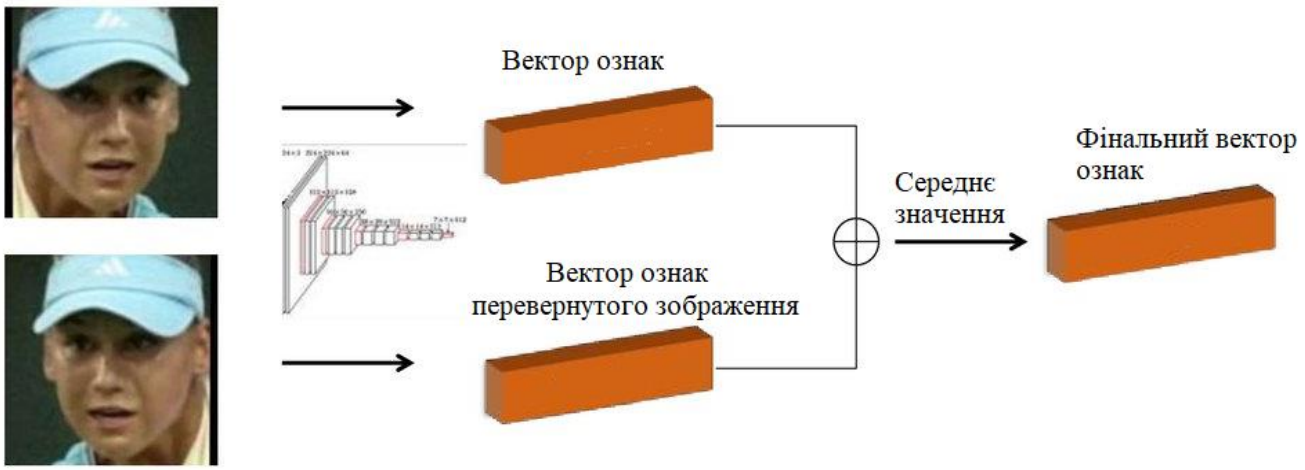

Рисунок 2.9 – Аугментація під час тестування

Очевидно, що аугментація під час тестування повинна збігатися з аугментацією тренування, але не обов’язково повністю. Тестова аугментація може бути «легшою», за тренувальну, тобто менше змінювати зображення.

Можливі аугментації, що часто застосовуються на практиці:

– вертикальний переворот;

- горизонтальний переворот;
- випадковий переворот на 90 градусів;
- вирізати випадковий фрагмент зображення;
- розмиття;
- зміна кольорової гами;
- зміщення зображення;
- спотворення сітки – нелінійна зміна зображення по ключовим точкам.

Також, існує багато стратегій агрегації отриманих результатів під час тестової аугментації:

- усереднення векторів ознак;
- усереднення отриманих ймовірностей;
- зважене усереднення (наприклад, вага вектору ознак початкового зображення може бути значно більшою, за вагу векторів аугментованих зображень).

Слід зазначити, що аугментацію під час тренування застосовують завжди, бо цей підхід стабільно покращує метрики мережі, залишається лише підібрати відповідні перетворення, спираючись на показники метрики якості.

3 АНАЛІЗ РЕЗУЛЬТАТІВ

Для проведення експериментів скористаємося фреймворком для глибокого навчання PyTorch [28] на мові Python3 [29]. Також, для подібних цілей дуже зручно користуватися оболонкою Jupyter [30], що значно полегшує візуалізацію отриманих результатів.

Запропоновані у попередньому розділі методи та архітектурні зміни стосуються класифікаційної частини мережі, тож потрібно обрати згорткову мережу, поверх якої буде знаходитися модифікований Arcface класифікатор.

Автори ArcFace застосовували архітектуру ResNet50 [31]. ResNet – згорткова нейрона мережа, основною особливістю якої є з'єднання швидкого доступу, сутність яких полягає у додатковому обхідному шляху сигналу повз згорткові шари (див. рис. 3.1).

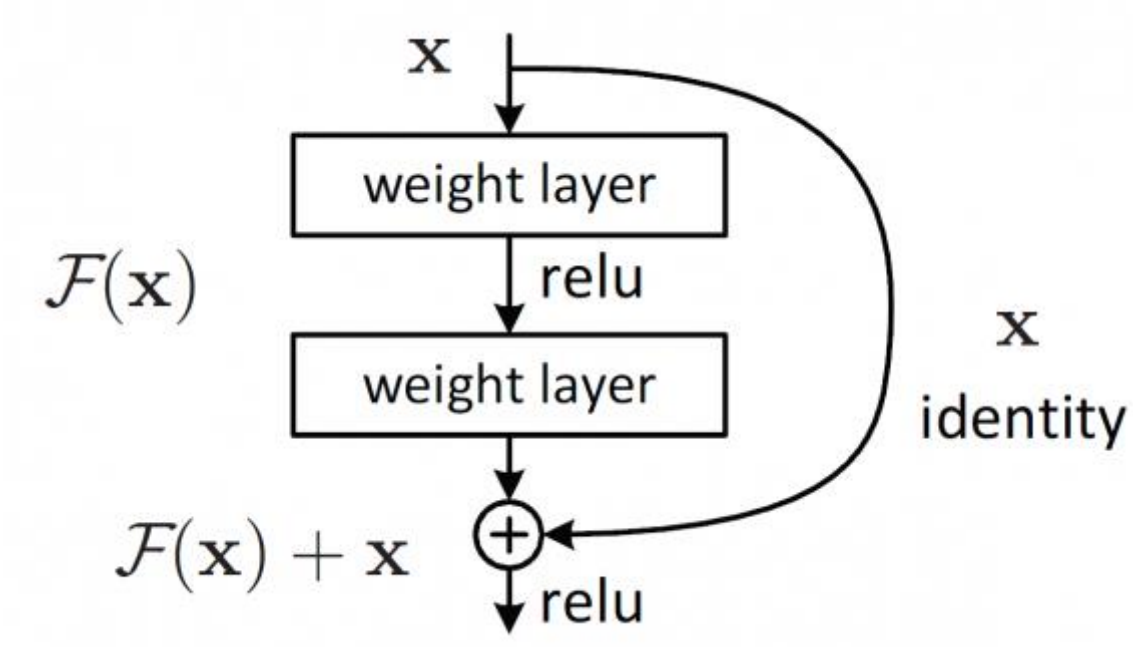

Рисунок 3.1 – З'єднання швидкого доступу

Основною перевагою є те, що таке з’єднання дозволяє боротися з затуханням градієнтів під час навчання глибоких мереж [32], тож така архітектура дозволяє

навчати значно глибші мережі, ніж класичні згорткові нейроні мережі. Архітектура ResNet складається з 4 блоків, загальну архітектуру яких зображено на рисунку 3.2.

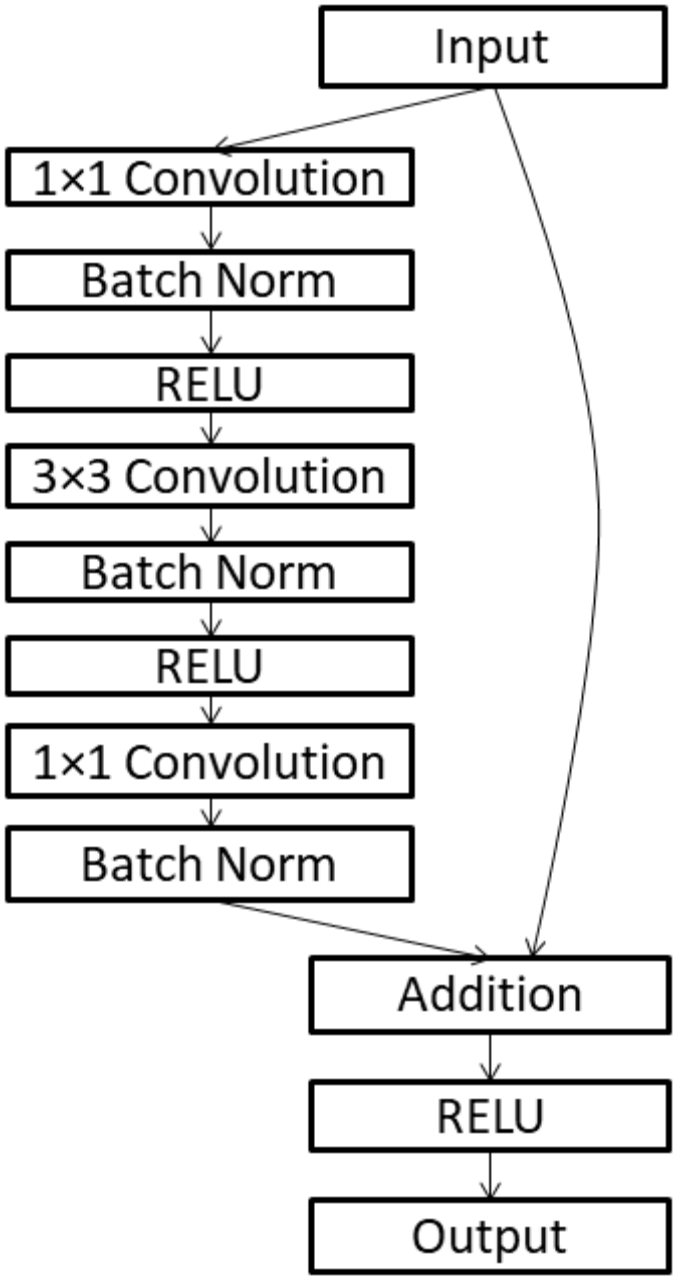

Рисунок 3.2 – Архітектура блоку ResNet

Код ініціалізації та інференсу загальної архітектури досліджуваної моделі зображено на рисунку 3.3.

```python
class ArcMarginHead(nn.Module):
  def __init__(self, num_classes, emb_size=512, easy_margin=False, margin_m=0.5, margin_s=30.0):
    super(ArcMarginHead, self).__init__()

    self.weight = torch.nn.Parameter(torch.FloatTensor(num_classes, emb_size))
    nn.init.xavier_uniform_(self.weight)

    self.easy_margin = easy_margin
    self.m = margin_m
    self.s = margin_s

    self.cos_m = math.cos(self.m)
    self.sin_m = math.sin(self.m)
    self.th = math.cos(math.pi - self.m)
    self.mm = math.sin(math.pi - self.m) * self.m

  def forward(self, input, label):
    x = F.normalize(input)
    W = F.normalize(self.weight)
    cosine = F.linear(x, W)
    sine = torch.sqrt(1.0 - torch.pow(cosine, 2))
    phi = cosine * self.cos_m - sine * self.sin_m  # cos(theta + m)
    if self.easy_margin:
        phi = torch.where(cosine > 0, phi, cosine)
    else:
        phi = torch.where(cosine > self.th, phi, cosine - self.mm)
    one_hot = torch.zeros(cosine.size(), device=device)
    one_hot.scatter_(1, label.view(-1, 1).long(), 1)
    output = (one_hot * phi) + ((1.0 - one_hot) * cosine)
    output *= self.s

    return output
```

Рисунок 3.3 – Код досліджуваної моделі

Аугментацію під час тренування буде застосовано на усіх варіантах архітектури, але ефективність аугментації під час тестування буде досліджена окремо через те, що далеко не у всіх випадках TTA значно підвищує метрики мережі, але затрачувані ресурси збільшуються у рази.

Для перетворення зображень застосуємо бібліотеку Albumentaion [33], що містить велику кількість різноманітних перетворень як для класифікації, так і для сегментації зображень. Також, перетворення у цій бібліотеці реалізовані значно швидше, відносно наївною Python реалізації, що дуже важливо при роботі з великими датасетами [34].

3.1 Триплетний відбір зразків

Результати перевірки ефективності різних стратегій триплетного відбору зразків на архітектурі ArcFace без змін представлені у таблиці 3.1.

Таблиця 3.1 – Порівняння стратегій триплетного відбору

	MegaFace, точність	CIFAR-10, f1-micro	Trillion-Pairs, FPR
без змін	0.775	0.863	0.848
випадкова	0.774	0.868	0.847
всі позитивні, всі негативні	0.772	0.866	0.849
найпростіший позитивний, найтяжчий негативний	0.778	0.865	0.841

Процес навчання при застосуванні усіх перерахованих стратегій та без них зображено на рисунку 3.4.

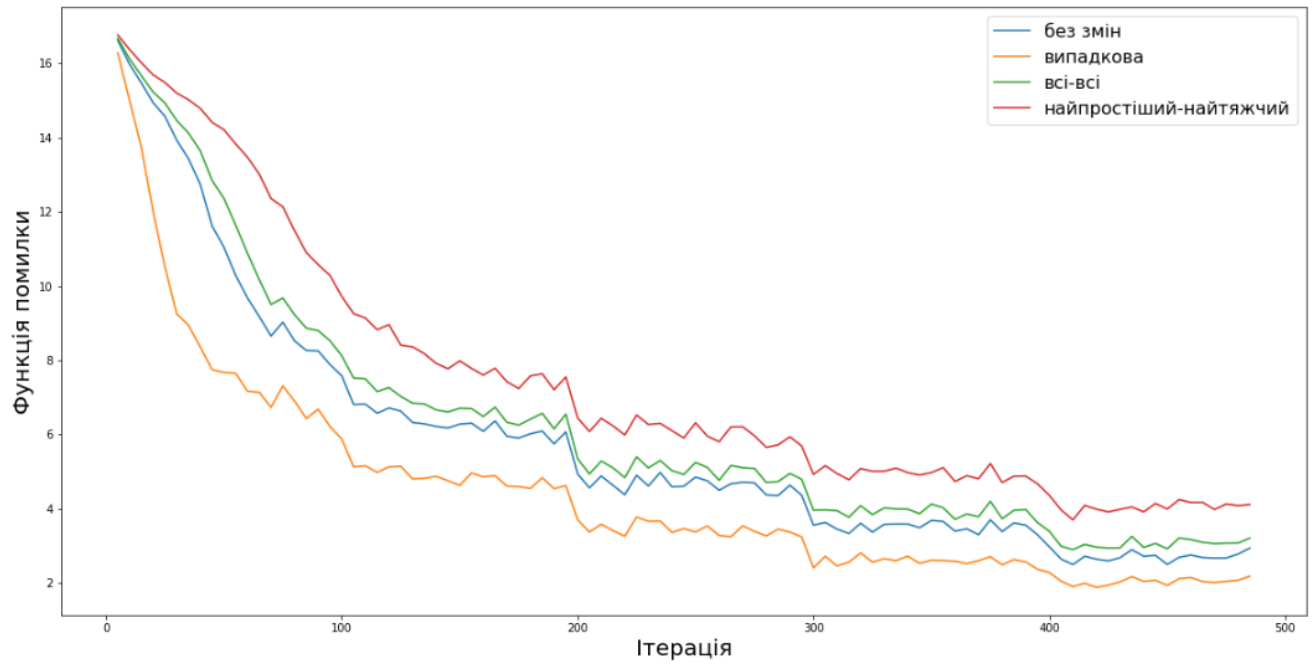

Рисунок 3.4 – Графік функції похибки з різними стратегіями триплетного відбору

Виходячи з результатів експериментів, триплетний відбір не дає значного приросту показників якості моделі. Можливо це пов’язано з тим, що ціллю триплетного відбору є досягнення внутрішньокласової компактності та збільшення межі класифікації, що й так досягається за допомогою косинусної границі ArcFace. Тож, даний підхід у рамках наших задач потребує багато ресурсів на створення пакетів відповідних даних, але не надає значущих переваг, тож скоріш за все використання триплетного відбору даних не має сенсу застосовувати з ArcFace-подібними мережами.

3.2 Нормалізація пакетів даних

У рамках цього експерименту додамо шар нормалізації між останнім згортковим шаром за останнім лінійним шаром, як показано на рисунку 3.5.

Рисунок 3.5 – Модифікація архітектури з шарами нормалізації

Результати застосування різних методів нормалізації представлені у таблиці 3.2.

Таблиця 3.2 – Порівняння стратегій нормалізації даних

	MegaFace, точність	CIFAR-10, f1-micro	Trillion-Pairs, FPR
без змін	0.775	0.863	0.848
нормалізація пакетів	0.787	0.885	0.860
нормалізація ваг	0.763	0.784	0.832
нормалізація шарів	0.776	0.873	0.850

Процес навчання при застосуванні усіх перерахованих стратегій нормалізації та без них зображено на рисунку 3.6.

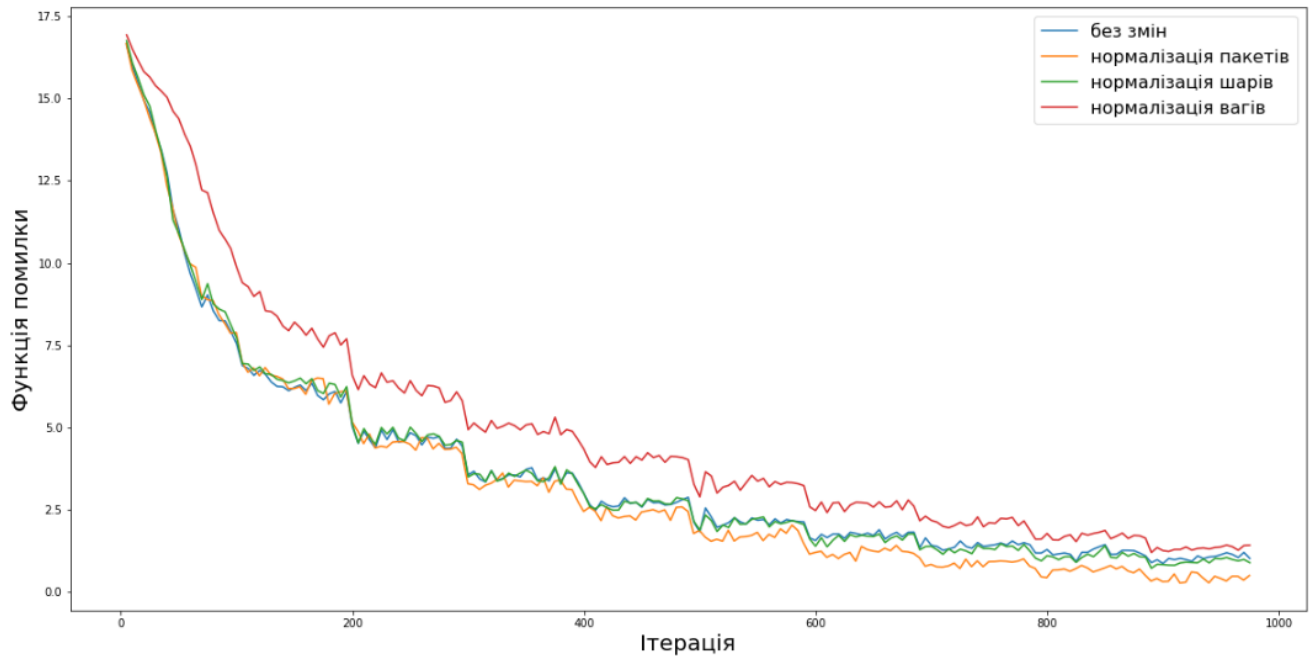

Рисунок 3.6 – Графік функції похибки з різними стратегіями нормалізації

Пакетна нормалізація покращує метрики якості на усіх задачах більше, ніж інші стратегії, також пришвидшує та стабілізує навчання.

3.3 Шар виключення

Шар виключення має сенс додати перед останнім лінійним шаром (див. рис. 3.7), завдяки чому мережа буде навчатися виділяти більш атомарні ознаки та зменшить перенавчання.

Рисунок 3.7 – Модифікація архітектури з шаром виключення

Результати роботи моделі з різними ймовірностями виключення представлені у таблиці 3.3.

Таблиця 3.3 – Порівняння параметру ймовірності для шару виключення

	MegaFace, точність	CIFAR-10, f1-micro	Trillion-Pairs, FPR
p=0	0.775	0.863	0.848
p=0.2	0.783	0.877	0.852
p=0.5	0.795	0.894	0.870
p=0.7	0,770	0.843	0.843

Процес навчання з шаром нормалізації різної ймовірності та без нього зображено на рисунку 3.8. На графіку бачимо, що виключення с ймовірністю 0.7 погіршує як процес навчання, так і кінцевий результат.

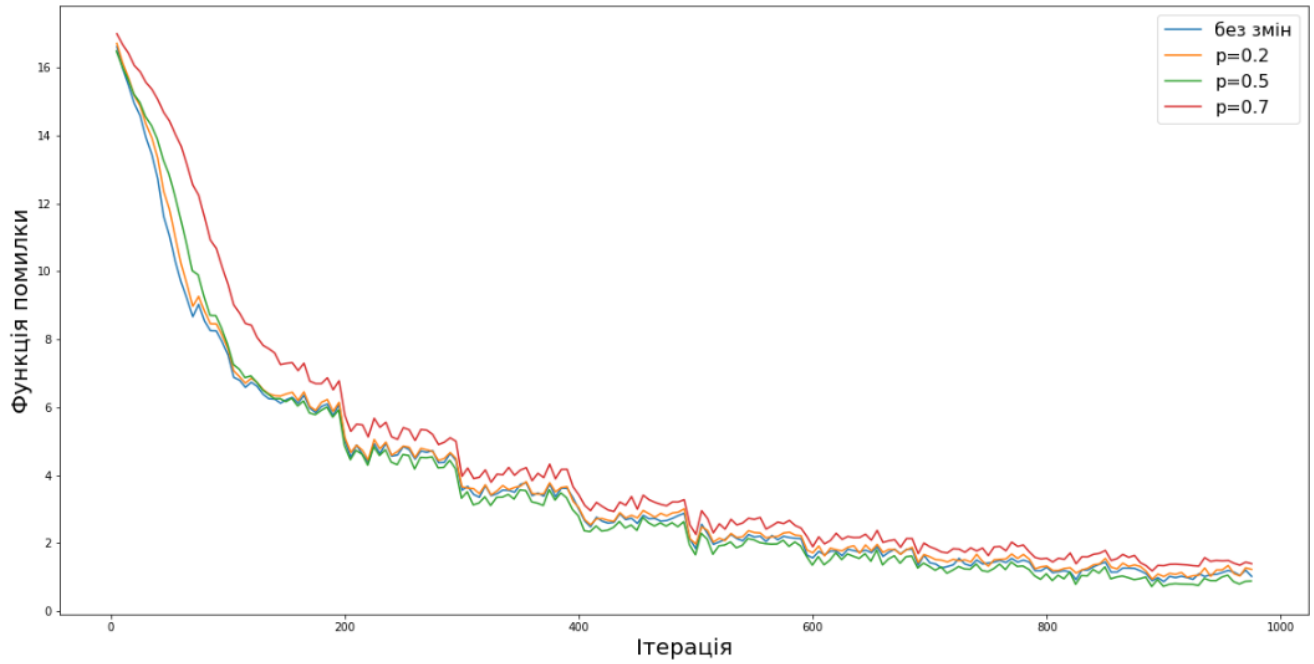

Рисунок 3.8 – Графік функції похибки з різними ймовірностями виключення

Шар виключення з ймовірністю p=0.5 показав найкращі результати, має сенс включити його до фінальної архітектури моделі.

3.4 Аугментація під час тестування

Для порівняння метрик якості розглянемо два варіанта аугментації: просту та складну. За просту візьмемо горизонтальний переворот, тобто інференс буде проводитися на оригінальному зображенні та його горизонтально перевернутій копії, результат будемо отримувати простим усередненням. В складну аугментацію будуть входити три перетворення: зміщення, масштабування та поворот. Усі

перетворення ймовірнісні, тобто те, наскільки зображення зміниться, буде коливатися у деякому проміжку.

Результати роботи моделі з різними TTA представлено у таблиці 3.4:

Таблиця 3.4 – Порівняння аугментацій під час тестування

	MegaFace, точність	CIFAR-10, f1-micro	Trillion-Pairs, FPR
без змін	0.775	0.863	0.848
горизонтальний поворот	0.774	0.861	0.851
складна аугментація	0.776	0.842	0.847

Як можна побачити, результати неоднозначні: інколи метрика йде вгору, інколи вниз. Накладні витрати від використання цього методу набагато більші, ніж отримана вигода, тож має сенс відмовитися від використання аугментації під час тестування у фінальній моделі.

3.5 Об’єднання методів

У попередніх розділах було з'ясовано, що шар виключення та шар нормалізації пакетів даних покращують метрики на поточних задача класифікації та реідентифікації зображень. Об’єднаємо ці підходи у єдину архітектуру та перевіримо ефективність відносно оригінальної ArcFace моделі.

Для шару виключення встановимо ймовірність 0.5, так як з таким параметром метод показав себе найкраще.

Найкращою стратегією нормалізації виявилася нормалізація пакетів даних, що не дивно, бо такий вид нормалізації є вибором за замовчуванням для більшості задач.

Архітектуру фінальної моделі зображено на рисунку 3.9.

Рисунок 3.9 – Схема фінальної архітектури

Код отриманою архітектури представлено на рисунку 3.10.

```
class FinalBottleneck(nn.Module):
  def __init__(self, x1=512, x2=512, p=0.5):
    super(FinalBottleneck, self).__init__()

    self.bn1 = nn.LayerNorm(512)
    self.do = nn.Dropout(p=p)
    self.fc = nn.Linear(in_features=x1, out_features=x2)
    self.bn2 = nn.LayerNorm(512)

  def forward(self, x):
    x = self.bn1(x)
    x = self.do(x)
    x = self.fc(x)
    x = self.bn2(x)

    return x
```

Рисунок 3.10 – Код фінальної архітектури

Процес навчання зображено на рисунку 3.11. З графіку бачимо, що модифікована версія на перших ітераціях показує результати гірші, за оригінал, але на останніх ітераціях покращує результат оригіналу.

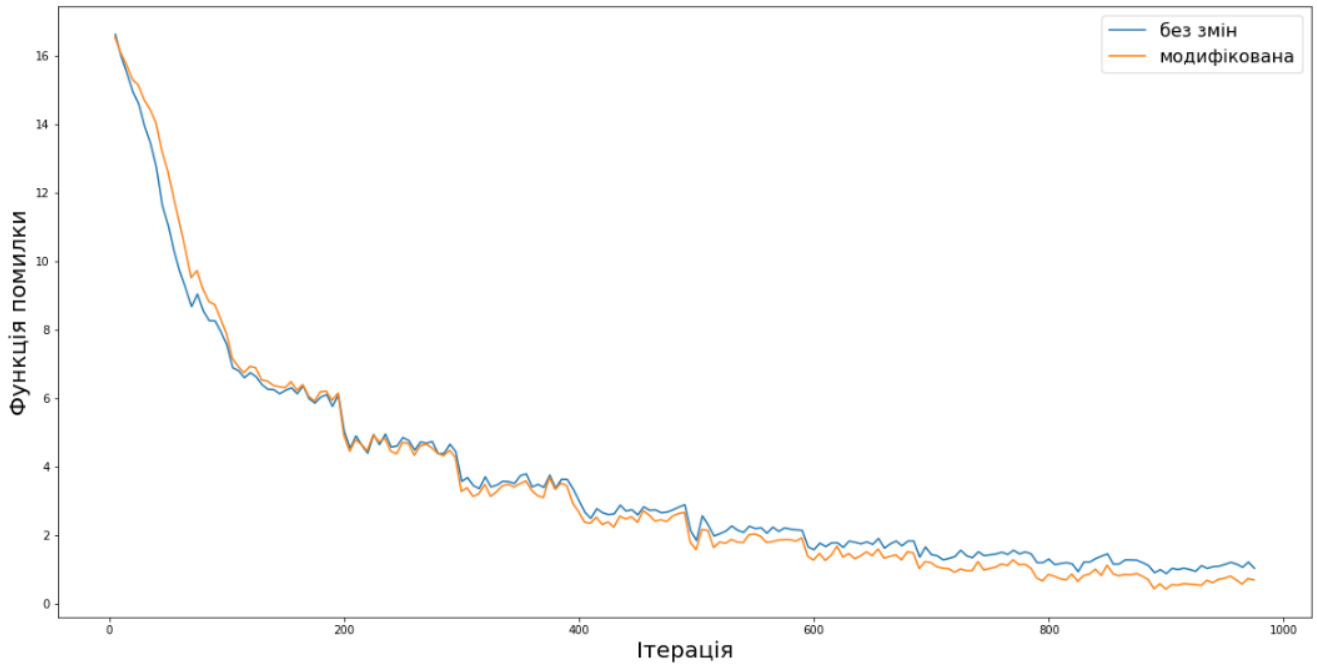

Рисунок 3.11 – Графік функції похибки початкової та фінальної архітектури

Результати порівняння оригінальної моделі та отриманої представлено у таблиці 3.5.

Таблиця 3.5 – Порівняння архітектур

	Класичний ArcFace	Модифікований ArcFace
MegaFace, точність	0.775	0.782
CIFAR-10, f1-micro	0.863	0.891
Trillion-Pairs, FPR	0.848	0.847
Час виконання інференсу	55.1 мс	59.2 мс

Отже, отримана модель покращує результати оригінальної моделі на задачах реідентифікації з відкритим та закритим датасетами, але програє у часі виконання інференсу, по причині додаткових шарів.

ВИСНОВОК

У результаті виконання атестаційної роботи магістра було проведено дослідження методів навчання ознак для вирішення задач реідентифікації об'єктів на зображеннях.

Було оглянуто патенту та наукову літературу за темою роботи. Було розглянуто класичну архітектуру згорткової нейронної мережі для класифікації та виявлено труднощі для її використання при вирішенні задачі реідентифікації: динамічне змінення кількості класів, мала вибірка, лінійна роздільність. В наслідок цього було вирішено використовувати метод навчання ознак. Було розглянуто сіамську нейронну мережу, нейронну мережу SphereFace, нейронну мережу ArcFace.

Було сформульовано наукову та науково-технічну задачу: опис та дослідження модифікацій ArcFace, реідентифікація з відкритим набором даних, реідентифікація з закритим набором даних, попарна реідентифікація та реалізація прикладної програмної системи.

Для оптимізації ArcFace було розглянуто та досліджено метод триплетного відбору зразків, нормалізацію пакетів даних, шар виключення та аугментацію зображень як під час тренування, так і під час тестування.

Отриманий у результаті дослідницький скрипт дозволяє зручно та швидко створювати архітектури нейронних мереж для навчання ознак, тестувати та візуалізувати результати їх роботи.

При виконанні атестаційної роботи було отримано модифіковану архітектуру ArcFace, що покращує метрики якості на задачах реідентифікації з відкритим та закритим датасетами.

Результати, що було отримано, є важливими для подальшого використання у науковій діяльності, бо отримана архітектура відкриває шляхи для подальшої оптимізації її гіперпараметрів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Електронний ресурс] URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 28.02.2020).

2) Wang H., Wang Y., Zhou Z., Ji X., Gong D., Zhou J., Li Z., and Liu W. CosFace: Large Margin Cosine Loss for Deep Face Recognition. arXiv preprint arXiv: 1801.09414, 2018.

3) Agarap A. Deep Learning using Rectified Linear Units (ReLU) arXiv preprint arXiv: 1803.08375, 2018.

4) Wang J., Zhou F., Wen S., Liu X. and Lin Y. Deep Metric Learning with Angular Loss. arXiv preprint arXiv: 1708.01682, 2017.

5) Utkin L., Kovalev M. and Kasimov E. An explanation method for Siamese neural networks. arXiv preprint arXiv: 1911.07702, 2019.

6) The MNIST database of handwritten digits [Електронний ресурс] URL: http://yann.lecun.com/exdb/mnist/ (дата звернення: 02.03.2020).

7) Liu W., Wen Y., Yu Z., Li M., Raj B., Song L. SphereFace: Deep Hypersphere Embedding for Face Recognition // [Матеріали конференції] Conference on Computer Vision and Pattern Recognition, 2017.

8) Deng J., Guo J., Xue N., Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. arXiv preprint arXiv: 1801.07698, 2018.

9) MegaFace and MF2: Million-Scale Face Recognition [Електронний ресурс] URL: http://megaface.cs.washington.edu/ (дата звернення: 01.04.2020).

10) The CIFAR-10 and CIFAR-100 dataset [Електронний ресурс] URL: https://www.cs.toronto.edu/~kriz/cifar.html (дата звернення: 07.04.2020).

11) A. Yerokhin ; O. Turuta ; A. Babii ; A. Nechyporenko. Intelligent information system of heterogeneous medical data analysis // [Матеріали конференції] 12th

International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 2017.

12) Trillionpairs [Електронний ресурс] URL: http://trillionpairs.deepglint.com/overview (дата звернення: 15.04.2020).

13) Zoph B., V. L. Quoc. Neural architecture search with reinforcement learning. . arXiv preprint arXiv: 1611.01578, 2017.

14) BiMorf [Електронний ресурс] URL: http://bimorf.tilda.ws/ (дата звернення: 20.04.2020).

15) Hoffer E., Ailon N. Deep metric learning using triplet network // [Матеріали конференції] International Workshop on Similarity-Based Pattern Recognition, 2015.

16) W. Chen, X. Chen, J. Zhang, and K. Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. // [Матеріали конференції] Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, 2017.

17) A. Hermans*, L. Beyer*, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. arXiv preprint arXiv:1703.07737, 2017.

18) Improved Embeddings with Easy Positive Triplet Mining [Електронний ресурс] URL: https://www.groundai.com/project/improved-embeddings-with-easy-positive-triplet-mining/1 (дата звернення: 22.04.2020).

19) Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift // [Матеріали конференції] International Conference on Machine Learning, 2015.

20) Weight Normalization and Layer Normalization Explained (Normalization in Deep Learning Part 2) Mining [Електронний ресурс] URL: https://mlexplained.com/2018/01/13/weight-normalization-and-layer-normalization-explained-normalization-in-deep-learning-part-2/ (дата звернення: 22.04.2020).

21) Salimans T., Kingma D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. arXiv preprint arXiv: 1602.07868, 2016.

22) J. L. Ba, J. R. Kiros, G. E. Hinton. Layer Normalization. arXiv preprint arXiv: 1607.06450, 2016.

23) B. Ghojogh, M. Crowley. The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. arXiv preprint arXiv: 1905.12787, 2019.

24) Labach A., Salehinejad H. Survey of Dropout Methods for Deep Neural Networks arXiv preprint arXiv:1904.13310, 2019.

25) S.Fort, P. K. Nowak, S. Jastrzebski, S. Narayanan. Stiffness: A New Perspective on Generalization in Neural Networks. arXiv preprint arXiv: 1901.09491, 2020.

26) J. Wang, L. Perez. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. arXiv preprint arXiv: 1712.04621, 2017.

27) B. Li, F. Wu, S. Lim, S. Belongie, K. Q. Weinberger. On Feature Normalization and Data Augmentation arXiv preprint arXiv: 2002.11102, 2020.

28) PyTorch [Електронний ресурс] URL: https://pytorch.org/ (дата звернення 24.04.2020).

29) P.Joshi. Artificial Intelligence with Python. A Comprehensive Guide to Building Intelligent Apps for Python Beginners and Developers, – Packt Publishing, 2017. – 448p.

30) Project Jupyter [Електронний ресурс] URL: https://jupyter.org/ (дата звернення 24.04.2020)

31) K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. arXiv preprint arXiv: 1512.03385, 2015.

32) F. Huang, J. Ash, J. Langford, R. Schapire. Learning Deep ResNet Blocks Sequentially using Boosting Theory. arXiv preprint arXiv: 1706.04964, 2017.

33) A. Buslaev, A. Parinov, E. Khvedchenya, V.I. Iglovikov, A.A. Kalinin, Albumentations: Fast and flexible image augmentations, arXiv preprint arXiv:1809.06839 (2018).

34) Yerokhin, A. ; Nechyporenko, A. ; Babii, A. ; Turuta, O. A new intelligence-based approach for rhinomanometric data processing // [Матеріали конференції] IEEE 36th International Conference on Electronics and Nanotechnology, ELNANO, 2016.