Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

FEW-SHOT LEARNING IN COMPUTER VISION

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення» - 121

Керівник курсової роботи

д.т.н., доц. Глибовець А.М.

_____

(підпис)

“____” _____ 2021 р.

Виконав студент Янкін І.С.

_____

(підпис)

“____” _____ 2021 р.

Київ-2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

_____

(прізвище та ініціали)

_____

(підпис)

“____” _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Янкіну Ігорю Сергійовичу

факультету інформатики 3 р.н. бакалаврської програми

ТЕМА: Few-shot learning in computer vision

Зміст ТЧ до курсової роботи:

Календарний план

Вступ

Розділ 1. Аналіз існуючих рішень та постановка завдання

Розділ 2. Аналіз існуючих методів

Розділ 3. Реалізація методів та аналіз отриманих результатів

Висновки

Список використаної літератури

Додатки (за необхідністю)

Дата видачі “____” _____ 2020 р. Керівник _____*(підпис)*

Завдання отримав _____*(підпис)*

Тема: Few-shot learning in computer vision

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	13.10.2020	
2.	Пошук інформаційних джерел за темою	01.11.2020	
3.	Ознайомлення зі знайденою літературою	01.01.2021	
4.	Обрання методів, які будуть розглядатися	15.01.2021	
5.	Теоретичні дослідження	01.03.2021	
6.	Реалізація обраних методів	01.05.2021	
7.	Аналіз отриманих результатів	05.05.2021	
8.	Написання текстової частини	16.05.2021	
9.	Створення презентації	16.05.2021	
10.	Завантаження роботи для перевірки на плагіат	17.05.2021	
11.	Захист роботи		

Студент Янкін І.С.

Керівник Глибовець А.М.

“___” _____

ЗМІСТ

АНОТАЦІЯ

У роботі розглянута проблематика вирішення задач з галузі машинного навчання за умови дуже обмеженої кількості зображень для тренування.

Проведено аналіз усіх існуючих розповсюджених методів вирішення цієї проблеми, які допомагають досягти задовольняючих результатів без потреби використовувати велику кількість зображень у якості прикладів для тренування. Виконана реалізація обраних методів та їх порівняння відповідно до визначених у роботі критеріїв.

Відповідно до отриманих результатів були зроблені висновки про переваги та недоліки реалізованих методів

Ключові слова: комп’ютерне бачення, машинне навчання, штучний інтелект, класифікація зображень, обмеженні данні

ВСТУП

Опис проблеми

Згідно з загальноприйнятим визначенням, машинне навчання - це підрозділ штучного інтелекту, що базується на вивченні комп'ютерних алгоритмів, які дозволяють комп'ютерним програмам автоматично вдосконалюватися завдяки досвіду. Найпоширеніші алгоритми з цієї галузі базуються на здатності алгоритму вдосконалити результати завдяки досвіду, отриманого з великої кількості тренувальних прикладів. Однак, для деяких задач неможливо або дуже складно отримати достатньо контрольованих даних, яких би вистачило для здобуття задовільного результату з використанням цих алгоритмів. Ці задачі є доволі поширеними у галузі машинного зору. Зокрема задачі класифікації, потребують зібрати багато зображень об'єктів шуканих класів, що не завжди можливо через конфіденційність даних, міркувань про безпеку та інших причин. Такі задачі відносяться до окремого розділу та мають спеціальні методи для їх вирішення.

Few-Shot learning – підгалузь машинного навчання, мета якої - вирішення проблем за умови обмеженої кількості контрольованих даних. Маючи необхідність скористатися одним з алгоритмів цієї галузі, перед користувачем встає проблема обрання алгоритму, який найкраще підходить для вирішення його задачі. Через те, що галузь з'явилась відносно недавно, кількість інформації про ці алгоритми, їх обмеження та найкращі умови для використання недостатня для свідомого обрання алгоритму. Також відсутні порівняння алгоритмів. Саме цю проблему планується вирішити у цій роботі.

Вирішення цієї проблеми дозволить покращити результати під час вирішення задач, які потребують використання цих алгоритмів, завдяки можливості обрати найбільш відповідний та ефективний алгоритм для даної

задачі. Також це дозволить краще зрозуміти недоліки та переваги алгоритмів, які будуть розглянуті.

Структура роботи

Робота складається з 3 розділів.

У першому розділі проводиться аналіз доступних рішень для зазначеної проблеми. Обираються найпоширеніші та найпопулярніші рішення для подальшого розгляду. Формуються основні критерії, за якими методи будуть порівнюватись.

Другий розділ містить теоретичні відомості про кожний з обраних методів, засоби їх реалізації, ключові відмінності та особливості.

У третьому розділі описаний хід виконання практичної роботи по порівнянню обраних методів, надаються отриманні результати та, відповідно до них, аналізуються переваги та недоліки кожного з методів по відношенню до інших.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз рішень

Few-shot learning – галузь, що з’явилась доволі недавно. Одні з перших робіт, що присвячені цій темі датуються 2005 та 2006 роком [1,2]. Відповідно до цього, галузь все ще розвивається та сформована не остаточно. Так лише за 2021 рік було написано більш ніж сто сімнадцять наукових робіт, що можна знайти у відкритому доступі, які були присвячені цій темі [3]. Незважаючи на те, що існує багато методів вирішення проблеми та постійно з’являються нові, у роботі будуть розглянуті лише ті з них, які є найбільш поширеними. До таких методів можна віднести наступні:

- Transfer learning
- Distance based learning
- Humiliation based learning
- Meta learning

Усі ці методи є добре дослідженими та їх можна реалізувати використовуючи популярні фреймворки для роботи з машинним навчанням, такі як TensorFlow або Pytorch. Також існують окремі готові реалізації цих методів з використанням вищезазначених фреймворків, що доступні у вільному доступі.

1.2 Обрання критеріїв для порівняння

Сформуємо критерії за якими у подальшому будемо порівнювати обрані методи. Віднесемо до цих критеріїв наступні:

- Точність
- Швидкість тренування
- Легкість масштабування
- Використання додаткових матеріалів

Під точністю будемо розуміти відношення правильно розпізнаних зображень до усіх спроб класифікації. Ця метрика має свої недоліки [4], але вона

є найбільш зрозумілою для користувачів і саме вона є найголовнішим фактором, на який спирається багато людей під час обрання методу.

Швидкістю тренування у роботі вважається час між стартом тренування та готовністю моделі для використання. Через те, що швидкість залежить від обладнання, при порівнянні буде враховуватися різниця між результатами у процентному відношенні. Уся робота виконується з використанням обладнання GPU Nvidia GeForce RTX 2060 Super.

Легкістю масштабування будемо називати послідовність дій, необхідну для додання нового класу до моделі та порівнювати за кількістю дій та часу, що вони займають у користувача.

До додаткових матеріалів входять набори даних, відмінні від цільового набору даних, та моделі, які можуть бути необхідними для реалізації певного методу. Під час порівняння буде розглядатись необхідність таких додаткових матеріалів та складність їх знаходження у вільному доступі.

Таким чином, на основі вищезазначених чотирьох критеріїв буде порівняно усі методи та в залежності від результатів сформовані висновки стосовно їх переваг та недоліків.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

2.1 Transfer learning

Transfer learning є одним із найпоширеніших методів. На відміну від класичного навчання, у якому кожна задача розглядається ізольовано та не впливає на інші, цей метод засновано на тому, щоб завдяки знанням отриманим з одного набіру данних, врахувати ці знання та перенести їх на схожий набір данних, що дозволяє значно пришвидшити процес навчання та потребує набагато менше зображень, ніж при звичайному навчанні.

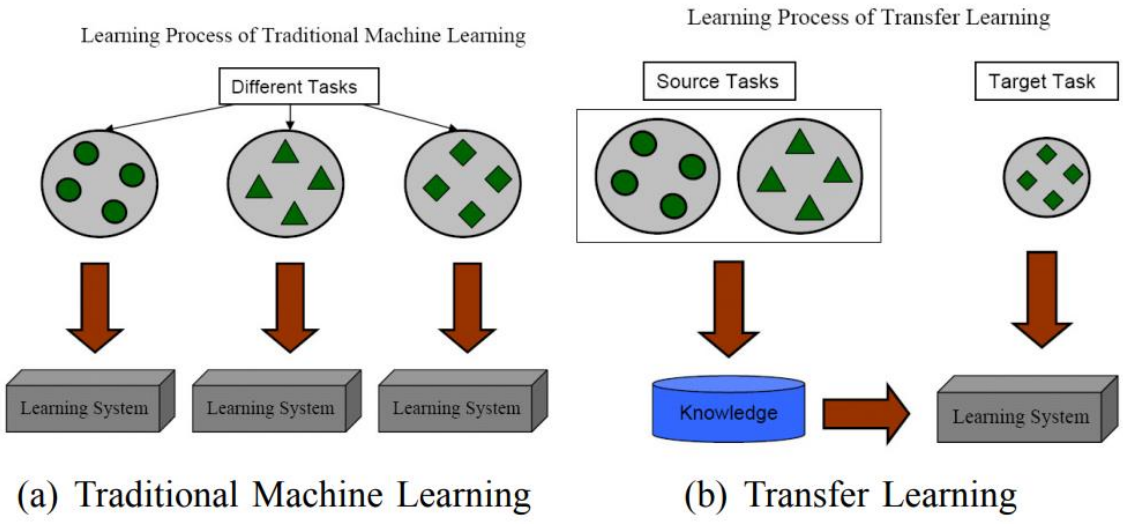

Рисунок 2.1.1 - Відмінності класичного машинного навчання (a) та transfer learning (b) [5]

Визначаючи формально, метод Transfer learning, маючи домен

$$D = \{x, P(X)\},$$

де x- простір особливостей;

$P(X)$ - відособлений розподіл з

$$X = \{x | x_i \in X, i = 1, \dots, n\};$$

вирішує задачу

$$T = \{y, f(x)\},$$

де y - набір можливіх результатів;

$f(x)$ - функція передбачення, яка отримується завдяки навчанню з пар $\{x_i, y_i\}, x_i \in x, y_i \in y$.

Задача вирішується для домену D_t завдяки покращенню функції $f(x)$, використовуючи для цього знання отриманні при виріщенні задачі T_s для домену $D_s, D_t \neq D_s$. У такому випадку D_s називається вихідним доменом, а D_t - цілівим доменом, при цьому розмір домену D_s значно більше, ніж розмір домену D_t [6].

Відповідно до загальноприйнятої класифікації, існують чотири різних підходи до реалізації цього методу [7]:

- Передача екземплярів
- Передача особливостей
- Передача параметрів
- Передача відношень

Передача екземплярів базується на допущені, що зображення з вихідного домену можна частково використовувати під час дотреновування моделі. Припускається, що частина зображень вихідного домену є корисною, але інша частина заважає вдалій класифікації. Таким чином, під час тренування відбувається зміна ваг моделі таким чином, щоб зменшити вплив зображень, що заважають та збільшити вплив тих, що є корисними [8].

Передача особливостей полягає у використанні перших n шарів попередньо натренованої моделі у якості перших шарів нової моделі. Ця ідея працює завдяки тому, що перші шари відносяться до загальних особливостей, а останні до особливостей, що притаманні лише об’єктам певного набору даних [9]. При реалізації до скопійованих натренованих шарів додаються один чи декілька кінцевих шарів та обираються шари, що будуть у подальшому

тренуватися. Вибір, які шари залишати відкритими для тренування можна зробити користуючись наступною ілюстрацією (рисунок 2.1.2).

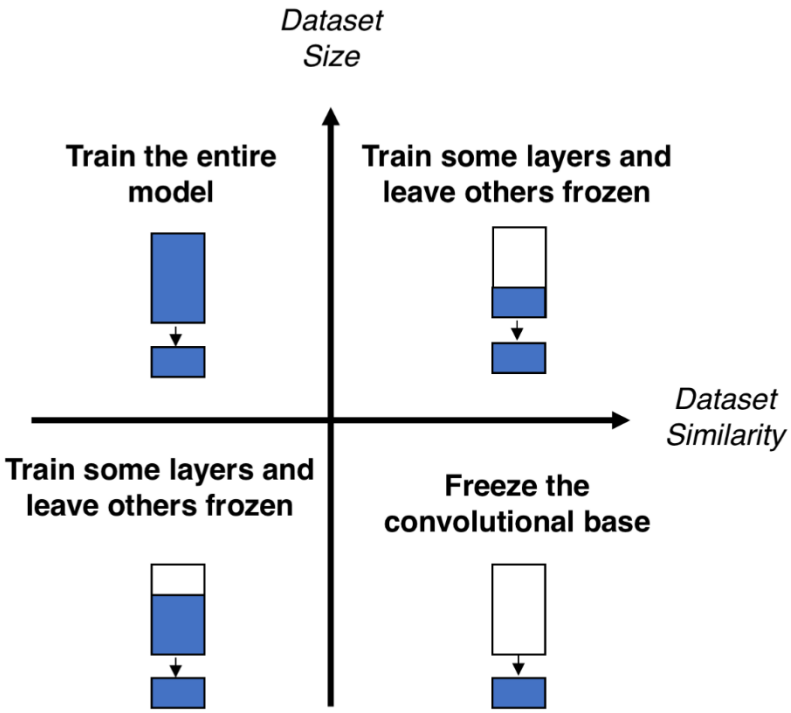

Рисунок 2.1.2 – Зображення простого алгоритму для обрання шарів для подальшого тренування [10]

Передача параметрів має як мету знаходження спільних параметрів між моделями вихідного та цільового доменів та подальше використання цих параметрів для вирішення задачі [11].

На відміну від інших технік, передача відношень використовується у випадках, коли данні мають певні зв’язки між собою. Таким чином, техніка концентрується на передачі саме цих відношень від вихідного домену до цільового. Прикладом алгоритмів, що реалізують цю техніку є TAMAR[12] та MLNs[13]

Слід зауважити, що у всіх вищезазначених техніках є недолік, який називається «негативна передача» (рисунок 2.1.3). Цей недолік логічно випливає з необхідності використовувати досвід отриманий під час вирішення іншої задачі

і полягає у тому, що при використанні доменів, що суттєво відрізняються один від одного, передача не тільки не покращує, а може й погіршити результати [14]. Тому для виростання цього методу дуже важливо мати правильний вихідний домен, що накладає певні обмеження на використання цього методу.

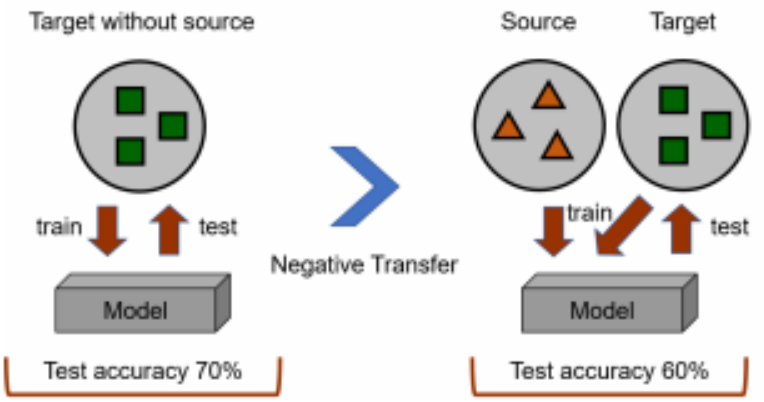

Рисунок 2.1.3 – Приклад негативної передачі [15]

Незважаючи на вищезазначений недолік, метод є дуже поширеним. У вільному доступі можна знайти різні попередньо навчені моделі, а більшість крупних фреймворків підтримують можливість донавчання моделі, що робить реалізацію цього методу простою та доступною для користувачів.

2.2 Distance based learning

У сфері машинного навчання широко розповсюдженні методи, засновані на відстанях. Ці методи базуються на можливості знайти чисельно виражену різницю між двома об’єктами. Для цього потрібно представити об’єкти у якості векторів чисел та застосувати одну з можливих функцій для здобуття відстані між ними.

Формально, методи засновані на відстанях, мають набір даних

$$\{x_i, y_i\}, 1 \leq i \leq L,$$

де x_i - вектор чисел розмірності n;

y_i – вихідний параметр, що може бути числом або бінарним кодом;

L – деяке ціле число, створюють набір з

$$r \leq L,$$

де r – деяке ціле число, зважених зразків

$$(u_i, v_i, w_i), u_i \in R^n, v_i \in N, w_i \in R,$$

де u_i – вхідні данні об’єкта;

v_i – класифікація об’єкта;

w_i – ваги об’єкта;

Тоді, маючи певний запит $q \in R^n$ та певну функцію обрахунку відстані між об’єктами d, результати для q залежатимуть лише від відстані між q до інших об’єктів [16].

Одна з найбільш поширених реалізацій метода базується на алгоритмі k-середніх [17]. Цей алгоритм проводить розбиття набіру данних на k кластерів (рисунок 2.2.1), об'єкти яких розташовані поруч один з одним. Маючи набір данних, ініціалізується k різних центрів випадковим чином. Далі для кожного об’єкту послідовно обирається найближчий кластер. Найближчим кластером вважається той, об’єктів якого більше всього серед k найближчих за функцією d відстаней. Після отримання найближчого кластеру, об’єкт вважається належним до цього кластеру, а його центр перераховується.

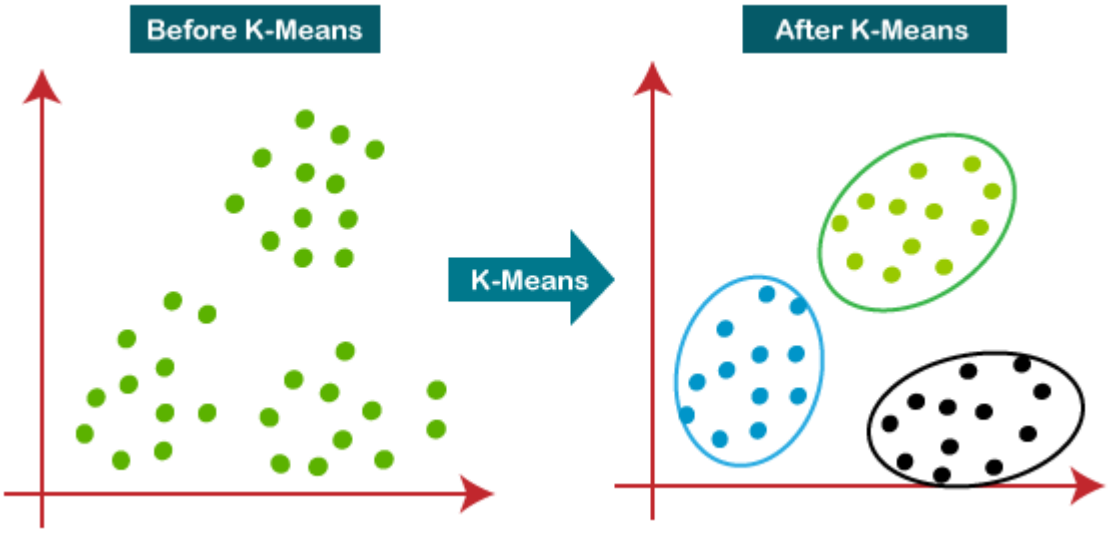

Рисунок 2.2.1 – Результат роботи алгоритму k-середніх [18]

Метод, що використовується у few-shot learning є певною модифікацію методу k-середніх. Для роботи він потребує попередньо збережені вектори особливостей для зображень цільового домену. Ці вектори утворюють k кластерів (рисунок 2.2.2), де k – кількість класів цільового домену. Далі за допомогою функції d визначається клас, до якого належить зображення.

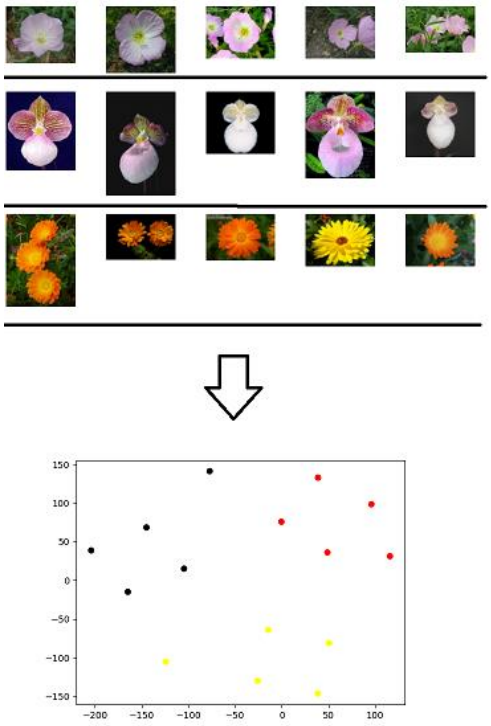

Рисунок 2.2.2 – Розбиття трьох класів на кластери

Дуже важливим для вдалої реалізації методу є функція відстані, що буде використовуватися. Зазвичай у ролі таких функцій виступають:

- Евклідова відстань
- Косинус подібності
- Манхеттенська відстань
- Відстань Геммінга
- Відстань Махаланобіса
- Відстань Мінковського

Зазвичай, саме евклідова відстань використовується і є найпопулярнішою, але інші функції працюють не гірше, а іноді і краще за неї [19], тому вибір функції є важливим кроком для реалізації методу.

Вагомою перевагою методу над іншими є легкість додавання нових класів, через те, що він не потребує тренування моделі

2.3 Humiliation based learning

Аугментація дуже корисна не лише для вирішення проблеми few-shot learning, але широко використовується у всій галузі. Ідея полягає у збільшені кількості даних для тренування шляхом деяких перетворень зображень. Це є ефективним рішенням, тому що більші набори даних призводять до кращих результатів під час тренування моделі [20].

Таким чином, метод, маючи задачу T та початковий набір даних

$$D_s = \{x_i, y_i\}, 1 \leq i \leq L,$$

де x_i – певне зображення;

y_i – клас зображення x_i,

L – деяке ціле число, створює новий набір даних

$$D_t = \{x_k, y_k\}, L \leq k \leq K,$$

де x_k – зображення, отримане шляхом застосування певної функції f до зображення x_i;

y_k- клас зображення x_k, що дорівнює класу зображення x_i;

K – деяке ціле число, який використовується для вирішення задачі T.

Виділяють три основні подходи аугментації, в залежності від типу функції f, яка використовується для перетворень:

- Геометрична
- Кольорова
- Аугментація з використанням машинного навчання

Геометрична аугментація заснована на геометричних функціях перетворення зображення. До них відносяться наступні перетворення:

- Обрізання зображення

- Поворот зображення
- Віддзеркалювання зображення
- Додавання шумів
- Масштабування

Кольорова аугментація перетворює кольори зображення. До кольорових відносяться такі перетворення, як:

- Зміна яскравості
- Зміна контрастності
- Зміна насиченості
- Зміна різкості
- Зміна балансу білого

Аугментація за допомогою машинного навчання базується на використанні моделей для генерації нових зображень, використовуючи існуючі.

Рисунок 2.3.1 – Приклади генерації зображень за допомогою моделі [20]

Зазвичай аугментація використовується не як окремий метод, а як доповнення до іншого методу, що дозволяє збільшити набір даних та

покращити результат. Одними з найефективніших перетворень є обрізання, віддзеркалювання, повороти та використання WGAN [21] [22].

Незважаючи на те, що аугментація покращує результати, при надмірній аугментації (рисунок 2.3.2) зображення може настільки відрізнятися від початкового, що його вже не можна буде віднести до класу початкового зображення, тому степінь аугментації повинна контролюватися.

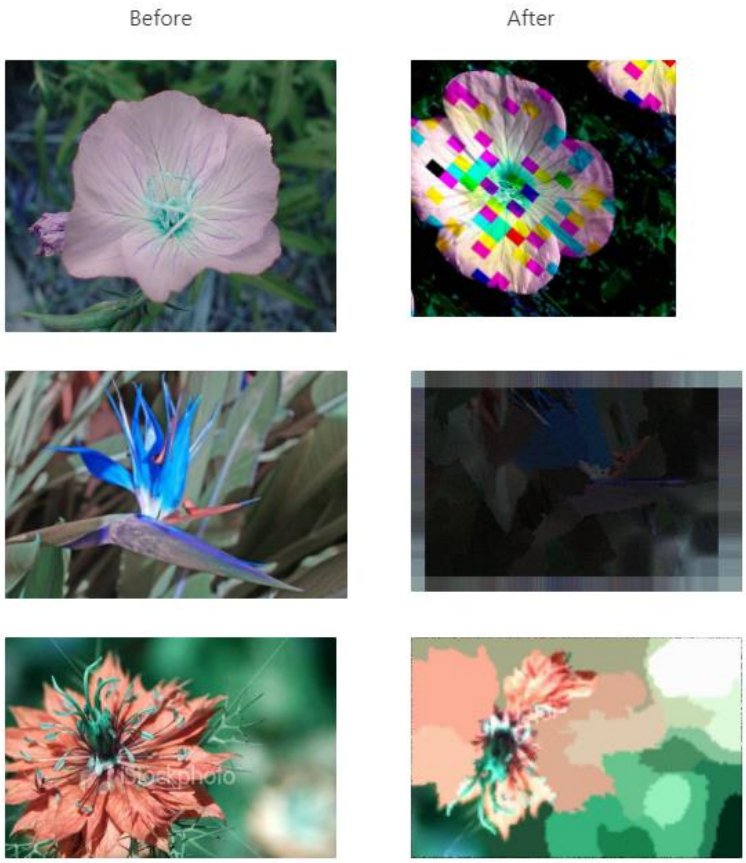

Рисунок 2.3.2 – Приклади надмірно аугментованих зображень

2.4 Meta learning

Мета навчання базується на використанні досвіду навчання, отриманого при вирішенні певних задач, у інших задачах. Таке навчання відрізняється від звичайної парадигми машинного навчання.

Визначимо мета навчання, як метод, який маючи вирішений набір задач $t_i \in T_s$, Вирішує нову задачу T_t, використовуючи досвід E, отриманий під час вирішення T_s.

Методи мета навчання можна поділити на три групи [23]:

- Навчання на оцінках моделей
- Навчання з використання мета особливостей
- Навчання з використанням моделей

Навчання на оцінках моделей використовує для навчання конфігурації моделей, отриманих під час вирішення задач з T_s. Такими конфігураціями можуть бути налаштування гіперпараметрів та компоненти архітектури. Реалізація методу полягає у тренуванні додаткової моделі, що буде передбачувати вдалу конфігурацію для вирішення нової задачі [24]. Цей метод може використовуватися без задач T_s. У цьому випадку відбувається K спроб навчання моделі з використанням випадкових конфігурацій для подальшого використання цих конфігурацій у якості T_s.

Навчання з використанням мета особливостей подає задачі з T_s як вектор особливостей

$$m(t_i) = (m_1, \dots, m_n).$$

Цей вектор використовується для подальшого визначення, яка задача найбільше схожа на задачу T_s, для подальшого використання досвіду отриманого під час вирішення цієї задачі для тренування моделі [25].

Останній з зазначених методів, навчання з використанням моделей, використовує моделі отриманні під час вирішення задач T_s для оптимізації моделі, що буде вирішувати задачу T_t. Зазвичай для оптимізації використовуються параметри моделей [26].

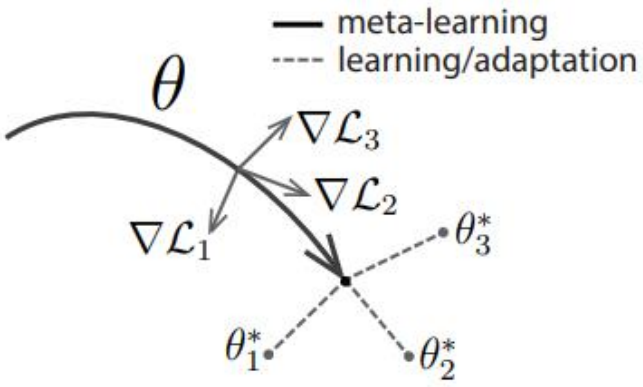

Рисунок 2.4.1 – Ілюстрація алгоритму MAML, що використовує ваги моделей для швидкої адаптації до нової задачі [27]

Одною з важливих переваг мета навчання можна вважати можливість покращуватись з часом. Так початковий набір задач, що було використано для навчання, може бути розширено додаванням нової задачі, що була з допомогою цього набору вирішено, що дозволяє покращувати подальший процес навчання [28].

3 РЕАЛІЗАЦІЯ МЕТОДІВ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Використані під час реалізації інструменти та данні

Для реалізації методів було використано різні датасети, за допомогою яких можна побачити особливості певних методів. Основним датасетом, що було використано для отримання результатів є датасет квітів, що складається з сто двох різних класів квітів [29]. З кожного класу було вибрано випадковим чином по п'ять зображень для формування датасету (додаток А), що демонструє проблему Few-shot learning. Більшість претренованих моделей, що використовуються були навчені на датасеті ImageNet [30], через його поширеність та наявність претренованих моделей на цьому датасеті для більшості доступних готових моделей. Також було використано датасет, що складається з трьохсот шістдесяти фруктів [31] та датасет, що складається зі ста чотирьох квітів [32] для демонстрації важливості правильного датасету для деяких з розглянутих методів.

Реалізація була написана на мові Python з використанням таких фреймворків для машинного навчання як Tensorflow2.0 та Pytorch. Ці фреймворки є одними з найпопулярнішими та надають користувачу багато можливостей, у тому числі можливість використання готових моделей зі списку, дотренування моделей та наявність деяких претренованих моделей. Також була використана бібліотека learn2learn [33, 34] заснована на фреймфорці Pytorch для реалізації мета навчання.

У якості моделі застосовується архітектура ResNet, яка є популярною для вирішення проблем комп'ютерного бачення [35].

3.2 Особливості реалізації методів

Для реалізації Transfer learning використовувались вбудовані в TensorFlow можливості використовувати попередньо навчену модель, видаляти та додавати нові шари, дозволяти або забороняти тренування для певних шарів (додаток Б)

Для реалізації Distance based learning було написано функцію, що обраховує косинус подібності між двома векторами (додаток В) та функцію, що визначає до якого класу належить зображення за відстанню між вектором зображення та векторами можливих класів (додаток Г). Для вилучення вектору особливостей з зображення, використовувались претреновані моделі з вилученим останнім шаром.

Для реалізації аугментації були написані функції перетворення, використовуючи бібліотеку imgaug, що дозволяє виконувати геометричні та кольорові перетворення. Були реалізовані функції, що проводять ці перетворення та на їх основі створено новий датасет для тренування (додатки Ґ, Д).

Для реалізації мета навчання було використано бібліотеку learn2learn [34] та з її допомогою реалізовано алгоритм MAML [27].

3.3 Порівняння моделей за отриманими результатами

При порівнянні методів за точністю (таблиця 3.3.1) можна побачити, що найефективнішим виявився Distance based learning, а на другому місці – Transfer learning. Ці два методи значно відриваються від результатів двох інших. Так мета навчання майже не покращило звичайне навчання моделі, а аугментація навіть погіршила ці результати.

Метод	Точність
Звичайне тренування	10%
Transfer learning	72%
Distance based learning	84%
Humiliation learning	5%
Meta learning	12%

Таблиця 3.3.1 – Результати точності методів

Примітка – до таблиці входить лише найкраща отримана точність, інші результати будуть розглянуті окремо.

Відповідно до результатів (таблиця 3.3.2), найшвидшим серед усіх є Distance based learning, тому що він не потребує навчання моделі. Незважаючи на це, можуть виникнути складності при роботі з багатьма класами, через те, що потребується вираховувати відстань до кожного з об’єктів. Найповільнішим виявився метод з використанням аугментацій, що обумовлено збільшеній кількістю зображень у датасеті. Інші методи швидше ніж звичайне тренування завдяки вагам, що адаптовані під виконання певної задачі.

Метод	Швидкість (с.)
Звичайне тренування	80
Transfer learning	44
Distance based learning	6
Humiliation learning	186
Meta learning	60

Таблиця 3.3.1 – Результати швидкості методів

Примітки:

1 До таблиці входить швидкість методу, за спроби, коли було отримано найкращий результат.

2 Швидкістю Distance based learning вважаємо час необхідний для отримання векторів з тренувальних зображень

За легкістю у масштабуванні Distance based learning є найпростішим у додаванні нових класів до моделей, тому що не потребує перетренування моделі. Достатньо зберегти зображення нового класу та вилучити з них особливості для подальшого порівняння. Усі інші методи потребують перенавчити модель для додавання нового класу, через це для інших методів будемо вважати легкість масштабування залежною від швидкості тренування моделі.

Так чи інакше, усі методи, окрім аугментації потребують додаткові матеріали для реалізації. Transfer learning потребує моделі, що попередньо натренована на схожому до цільового датасеті, Distance based learning зазвичай потребує моделі, яка буде вилучати вектор особливостей з зображення, а мета навчання потребує декілька різних моделей або датасетів для покращення процесу навчання.

3.4 Особливості окремих методів

Transfer learning було протестовано, використовуючи моделі, натреновані на різних датасетах, у якості претренованих. Так можна побачити (таблиця 3.3.3), що правильний датасет дуже важливий для вдалого дотренування. Чим менш схожий вихідний датасет, тим гірше результати тренування. Якщо вихідний датасет зовсім не схожий на цільовий (як у випадку з фруктами), результат може бути гіршим, ніж за звичайного тренування моделі. Також можна побачити, що тренування усіх шарів показує трохи кращі результати, ніж тренування лише останнього шару.

Датасет (шари, що тренувалися)	Точність
Фрукти (усі шари)	1%
Фрукти (останній шар)	1%
Інші квіти (усі шари)	72%
Інші квіти (останній шар)	68%
ImageNet (усі шари)	35%
ImageNet (останній шар)	26%

Таблиця 3.3.3 – Повні результати методу Transfer learning

Для методу Distance based learning, модель, що використовується для вилучення особливостей є не менш важливою, ніж претренована модель у Transfer learning. Погіршення результатів при неправильно обраній моделі дуже суттєве (таблиця 3.3.4), аж до отримання результатів гірше, ніж звичайне тренування.

Датасет	Точність
Фрукти	3%
Інші квіти	84%
ImageNet	33%

Таблиця 3.3.4 – Повні результати методу Distance based learning

Незважаючи на те, що аугментація показала найгірший результат точності і швидкості серед усіх, вона має зміст при сумісному використані з іншими методами. Так при використані разом з Transfer learning, результати були покращені на 3% та досягли 75%. Але з деякими методами аугментація дає негативний результат. Так при вилученні особливостей з аугментованого датасету під час реалізації Distance based learning, результат погіршився на 22% та був майже рівним зі звичайним тренуванням. Також, надмірна аугментація, як якісна, так і кількісна, може погіршувати результати (таблиця 3.3.5).

Сильна/слабка аугментація	Кількість аугментованих зображень	Точність
Сильна	5	3%
Слабка	5	5%
Слабка	20	3%

Таблиця 3.3.5 – Повні результати методу Humiliation learning

Мета навчання показало результати трохи краще та швидше, ніж звичайне. Нажаль, для демонстрації вдалої роботи мета навчання потребує багато різних даних. Так при використанні у мета навчені лише одного додаткового датасету квітів, навчання моделі було значно гіршим та недосягло навіть 1%. Лише при використані усіх з додаткових датасетів, вийшло покращити результат до 12%.

3.5 Переваги і недоліки методів

До переваг Transfer learning належить висока точність моделі та швидкість вище, ніж звичайна. Також, дуже суттєвою перевагою для користувача може стати легкість імплементації цього методу, так как він наявний в усіх великих фреймворках машинного навчання та є можливість найти велику кількість претренованих моделей. До недоліків відноситься залежність від претренованованої моделі та необхідність донавчати нову модель.

До переваг Distance based learning можна віднести точність та швидкість. Також вагомою перевагою є можливість з легкістю додати новий клас та відсутність необхідності навчати нову модель. Як недоліки можна зазначити лише відсутність готової реалізації у крупних фреймворках, через що користувачу потребується власноруч реалізувати деякі функції для роботи цього методу.

Аугментація, згідно з результатами, є невдалим рішенням для самостійного використання, проте може використовуватися разом з іншими методами. Головним недоліком аугментації є збільшення часу, необхідного для тренування та важливість правильної реалізації. При надмірному аугментувані результати погіршуються, що призводить до потреби правильно обирати функції перетворення та їх кількість.

Мета навчання влучно використовувати при наявності великої кількості даних. Як перевагу, можна відзначити можливість перевикористання моделі. Так модель, що була навчена за допомогою мета навчання, може бути використана для вирішення наступної задачі за допомогою мета навчання. Однак, якщо даних недостатньо, то покращення не буде помітне.

3.6 Висновки

Відповідно до результатів аналізу можна дійти висновку, що у загальному варіанті Transfer learning та Distance based learning є найбільш вдалим вибором для користувача. Аугментація має зміст лише як додаток до якогось методу, а не

як самостійний метод та потребує обережного використання. Мета навчання підходить при наявності великої кількості даних або коли планується багато задач, які можна буде використати для подальшого покращення.

ВИСНОВКИ

Відповідно до поставленої задачі, у роботі було сформовано список методів для огляду та критерії їх порівняння. Усі ці методи було класифіковано, сформовано формальне визначення та проаналізовано принципи їх роботи. Кожний метод було реалізовано та протестовано на одному й тому ж датасеті для подальшого порівняння отриманих результатів. Деякі методи були протестовані в різних умовах для того, щоб краще розкрити особливості, притаманні цим методам. Відповідно до отриманих результатів, методи були порівняні та було складено список переваг та недоліків кожного методу.

Ця робота має великий потенціал для продовження та доповнення. Незважаючи на те, що були розглянуті основні методи, що використовуються, залишається необхідність розглянути усі їх варіанти відповідно до зазначеної у роботі класифікації, для формування більш повного порівняння. Також, незважаючи на проведене порівняння, формування переваг та недоліків методів, залишається можливим провести роботу по створенню алгоритму для обрання певного методу для вирішення певної задачі.

Список використаної літератури

1. One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence [Електронний ресурс] / L. Fei-Fei, R. Fergus, P. Perona // 2006. – ст. 594–611 – Режим доступу: https://authors.library.caltech.edu/5407/1/LIFieeetpam06.pdf
2. Object classification from a single example utilizing class relevance metrics. In Advances in Neural Information Processing Systems [Електронний ресурс] / M. Fink // 2005. – ст. 449–456 – Режим доступу: https://www.researchgate.net/publication/221619654_Object_Classification_from_a_Single_Example_Utilizing_Class_Relevance_Metrics
3. Результати пошуку на сайті arxiv.org [Електронний ресурс] // Режим доступу: https://arxiv.org/search/advanced?advanced=&terms-0-operator=AND&terms-0-term=few-shot+learning&terms-0-field=title&classification-physics_archives=all&classification-include_cross_list=include&date-filter_by=specific_year&date-year=2021&date-from_date=&date-to_date=&date-date_type=submitted_date&abstracts=show&size=50&order=-announced_date_first&start=0
4. Classification: Accuracy [Електронний ресурс] // Режим доступу: https://developers.google.com/machine-learning/crash-course/classification/accuracy
5. A Survey on Transfer Learning [Електронний ресурс] / Sinno Jialin Pan, Qiang Yang Fellow // 2009. – ст. 2 – Режим доступу: https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf
6. A Survey on Deep Transfer Learning [Електронний ресурс] / Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu // 2018. – ст. 2 – Режим доступу: https://arxiv.org/pdf/1808.01974.pdf

7. A Survey on Transfer Learning [Електронний ресурс] / Sinno Jialin Pan, Qiang Yang Fellow // 2009. – ст. 5–6 – Режим доступу: https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf
8. Instance-Based Learning Algorithms [Електронний ресурс] / David W. Aha, Dennis Kibler, Marc K. Albert // 1991. – ст. 37–66 – Режим доступу: https://www.researchgate.net/publication/220343419_Instance-Based_Learning_Algorithms
9. Deep Learning of Representations for Unsupervised and Transfer Learning [Електронний ресурс] / Yoshua Bengio // 2012. – ст. 17–37 – Режим доступу: http://proceedings.mlr.press/v27/bengio12a/bengio12a.pdf
10. Transfer learning from pre-trained models [Електронний ресурс] / Pedro Marcelino // 2018. – Режим доступу: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751
11. Knowledge transfer via multiple model local structure mapping [Електронний ресурс] / J. Gao, W. Fan, J. Jiang, and J. Han // 2008. – ст. 283–291 – Режим доступу: https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=1306&context=sis_research
12. Mapping and revising markov logic networks for transfer learning [Електронний ресурс] / L. Mihalkova, T. Huynh, and R. J. Mooney // 2007. – ст. 608–614 – Режим доступу: https://www.aaai.org/Papers/AAAI/2007/AAAI07-096.pdf
13. Markov logic networks [Електронний ресурс] / M. Richardson, P. Domingos // 2006. – ст. 2–3 – Режим доступу: https://homes.cs.washington.edu/~pedrod/papers/mlj05.pdf
14. Characterizing and Avoiding Negative Transfer [Електронний ресурс] / Zirui Wang, Zihang Dai, Barnabas P ´ oczos, Jaime Carbonell // 2019. – ст. 1–9 –

Режим доступу: https://arxiv.org/pdf/1811.09751.pdf

15. Overcoming Negative Transfer: A Survey [Електронний ресурс] / Wen Zhang, Lingfei Deng, Lei Zhang, Dongrui Wu // 2020. – ст. 1 – Режим доступу: https://arxiv.org/pdf/2009.00909.pdf
16. A study of distance-based machine learning algorithms [Електронний ресурс] / Dietrich Wettschereck // 1994. – ст. 9–10 – Режим доступу: https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/zw12z7835
17. Some methods of classification and analysis of multivariate observations [Електронний ресурс] / J. MacQueen // 1967. – ст. 281-293 – Режим доступу: https://www.academia.edu/1261479/Some_methods_for_classification_and_analysis_of_multivariate_observations
18. K-означає кластеризацію [Електронний ресурс] // Режим доступу: https://ichi.pro/ru/k-oznacaet-klasterizaciu-66942525093873
19. Comparative Study of Distance Functions for Nearest Neighbors [Електронний ресурс] / Janett Walters-Williams, Yan Li // 2010. – ст. 83–84 – Режим доступу: https://www.researchgate.net/publication/221231823_Comparative_Study_of_Distance_Functions_for_Nearest_Neighbors
20. A survey on Image Data Augmentation for Deep Learning [Електронний ресурс] / Connor Shorten, Taghi M. Khoshgoftaar // 2019. – ст. 1–44 – Режим доступу: https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-019-0197-0.pdf 1-5
21. Wasserstein GAN [Електронний ресурс] / Martin Arjovsky, Soumith Chintala, and L´eon Bottou // 2017. – ст. 1–28 – Режим доступу: https://arxiv.org/pdf/1701.07875.pdf

22. Research on data augmentation for image classification based on convolutional neural networks [Електронний ресурс] / Jia S., Wang P., Jia P., Hu S. // 2017. – ст. 4165-4170 – Режим доступу: https://www.researchgate.net/publication/322282188_Research_on_data_augmentation_for_image_classification_based_on_convolution_neural_networks

23. Meta-Learning: A Survey [Електронний ресурс] / Joaquin Vanschoren // 2018. – ст. 2-16 – Режим доступу: https://arxiv.org/pdf/1810.03548.pdf

24. Dataset2Vec: Learning Dataset Meta-Features [Електронний ресурс] / Hadi S. Jomaa, Lars Schmidt-Thieme, Josif Grabocka // 2021. – ст. 5-7 – Режим доступу: https://arxiv.org/pdf/1905.11063.pdf

25. Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels [Електронний ресурс] / Massimiliano Patacchiola, Jack Turner, Elliot J. Crowley, Michael O'Boyle, Amos Storkey // 2020. – ст. 3-6 – Режим доступу: https://arxiv.org/pdf/1910.05199.pdf

26. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks [Електронний ресурс] / Chelsea Finn, Pieter Abbeel, Sergey Levine // 2017. – ст. 3-6 – Режим доступу: https://arxiv.org/pdf/1703.03400.pdf

27. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks [Електронний ресурс] / Chelsea Finn, Pieter Abbeel, Sergey Levine // 2017. – ст. 2 – Режим доступу: https://arxiv.org/pdf/1703.03400.pdf

28. Reconciling meta-learning and continual learning with online mixtures of tasks [Електронний ресурс] / Ghassen Jerfel, Erin Grant, Thomas L. Griffiths, Katherine Heller [Електронний ресурс] // 2019. – ст. 5–9 – Режим доступу: https://arxiv.org/pdf/1812.06080.pdf

29. 102 Flowers Diff Species DataSet [Електронний ресурс] // Режим доступу: https://www.kaggle.com/lenine/flower-102diffspecies-dataset

30. ImageNet [Електронний ресурс] // Режим доступу: https://www.image-net.org/index.php

31. Fruits 360 [Електронний ресурс] // Режим доступу: https://www.kaggle.com/moltean/fruits

32.104 Flowers: Garden of Eden [Електронний ресурс] // Режим доступу: https://www.kaggle.com/msheriey/104-flowers-garden-of-eden

33. learn2learn: A Library for Meta-Learning Research [Електронний ресурс] / Sébastien M. R Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, Konstantinos Saitas Zarkias // 2020. – ст. 1-8 - Режим доступу: https://arxiv.org/pdf/2008.12284.pdf

34. learn2learn [Електронний ресурс] // Режим доступу: https://github.com/learnables/learn2learn

35. Deep Residual Learning for Image Recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // 2015. – ст. 1-12 - Режим доступу: https://arxiv.org/pdf/1512.03385.pdf

36.COCO dataset [Електронний ресурс] // Режим доступу: https://cocodataset.org

Додаток А

(довідниковий)

Вибір п’яти випадкових зображень з датасету

```
from pathlib import Path
from random import shuffle
import shutil

folder = Path('datasets/flower_data/train')
mini_f = Path('datasets/flower_data/fewshot')
mini_f.mkdir(exist_ok=True, parents=True)

for pid in folder.glob('*'):
    imgs = list(pid.glob('*'))
    shuffle(imgs)
    imgs = imgs[:5]
    for im in imgs:
        (mini_f/im.parent.name).mkdir(exist_ok=True, parents=True)
        shutil.copy(im, mini_f/im.parent.name/im.name)
```

Додаток Б

(довідниковий)

Демонстрація налаштування претренованої моделі для подальшого навчання

```
model = keras.models.load_model('flowers_train.h5')
model.layers.pop()
flattened = tf.keras.layers.Flatten()(model.output)
fc2 = tf.keras.layers.Dense(len(os.listdir(train_data_dir)),
activation='softmax', name="act")(flattened)
model = tf.keras.models.Model(inputs=model.input, outputs=fc2)
for layer in model.layers[:-1]:
    layer.trainable = False
```

Додаток В

(довідниковий)

Функція знаходження косинуса подібності

```
def cosine_distance(x, y):
    epsilon = 1e-8
    x_norm = np.sqrt((x**2).sum(1, keepdims=True))
    y_norm = np.sqrt((y**2).sum(1, keepdims=True))
    x = x / (x_norm + epsilon)
    y = y / (y_norm + epsilon)
    product = np.expand_dims(x, 1) * y  # shape [n, m, c]
    cos = product.sum(2)  # shape [n, m]
    return 1.0 - cos
```

Додаток Г

(довідниковий)

Функція знаходження найбільш вірогідного класу

```
def get_res(features, classes_features):
    min = 1
    r = -1
    for k, v in classes_features.items():
        dist = np.average(cosine_distance(np.array([features]), np.array(v)))
        if dist < min:
            r = k
            min = dist
    return r
```

Додаток Ґ
(довідниковий)
функція простого перетворення

```
seq = aug.Sequential(
    [
        aug.Fliplr(0.2),
        aug.Flipud(0.1),
        smt(aug.Affine(
            scale={"x": (0.9, 1.1), "y": (0.9, 1.1)},
            translate_percent={"x": (-0.1, 0.1), "y": (-0.1, 0.1)},
            rotate=(-30, 30),
        )),
    ],
    random_order=True
)
```

Додаток Д
(довідниковий)
функція складного перетворення

```
seq_hard = aug.Sequential(
    [
        # geometric
        aug.Flipud(0.2),
        aug.Fliplr(0.4),
        smt(aug.CropAndPad(
            percent=(-0.05, 0.1),
            pad_mode=ia.ALL,
            pad_cval=(0, 255)
        )),
        smt(aug.Affine(
            scale={"x": (0.7, 1.3), "y": (0.7, 1.3)},
            translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)},
            rotate=(-45, 45),
            shear=(-16, 16),
            order=[0, 1],
            cval=(0, 255),
            mode=ia.ALL
        )),
        aug.SomeOf((0, 5),
            # noise
                   [
            aug.OneOf([
                aug.GaussianBlur((0, 3.0)),
                aug.AverageBlur(k=(2, 7)),
            ]),
            aug.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5)),
            aug.Emboss(alpha=(0, 1.0), strength=(0, 2.0)),
            aug.SimplexNoiseAlpha(aug.EdgeDetect(alpha=(0.5, 1.0))),
            aug.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.05*255),
per_channel=0.5),
            aug.OneOf([
                aug.Dropout((0.01, 0.1), per_channel=0.5),
                aug.CoarseDropout((0.03, 0.15), size_percent=(0.02, 0.05),
per_channel=0.2),
            ]),
            # color
            aug.Multiply((0.5, 1.5), per_channel=0.5),
            aug.Add((-5, 5)),
            aug.AddToHueAndSaturation((-10, 10)),
            aug.LinearContrast((0.5, 2.0), per_channel=0.5),
            smt(aug.ElasticTransformation(alpha=(0.5, 3.5), sigma=0.25)),
            smt(aug.PiecewiseAffine(scale=(0.01, 0.05))),
            smt(aug.PerspectiveTransform(scale=(0.01, 0.1)))
        ],
            random_order=True
        )
    ],
    random_order=True
)
```