МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

СТЕПАНИК Олександр Володимирович

Метод ідентифікації ознак тварин на основі глибокого навчання / Method for animal feature identification based on deep learning

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
О. В. Степаник
_____

Науковий керівник:
к.т.н., доцент Х. В. Ліп'яніна-Гончаренко
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ - 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
СТЕПАНИК Олександр Володимирович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод ідентифікації ознак тварин на основі глибокого навчання / Method for animal feature identification based on deep learning
керівник роботи к.т.н., доцент Х.В. Ліп'яніна-Гончаренко
затверджені наказом по університету від 20 грудня 2024 року № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- огляд предметної області;
- аналіз алгоритмів для розв'язку задач класифікації;
- аналіз нейронних мереж для вирішення проблем задач класифікації;
- постановка задачі дослідження;
- дослідження алгоритмів дрібнозернистої класифікації;
- запропонований алгоритм дрібнозернистої класифікації видів птахів;
- дослідження засобів реалізації запропонованого алгоритму;
- реалізація алгоритму класифікації видів птахів на основі глибокого навчання;
- результати експериментальних досліджень та порівняння з відомими рішеннями.

5. Перелік графічного матеріалу у роботі
- архітектура запропонованого гібридного методу;
- приклади зображень використаних для навчання;
- графіки результатів роботи запропонованого гібридного методу.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 1.12.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2025 р.	

Студент _____ О.В. Степаник

підпис

Керівник роботи _____ к.т.н., доцент Х. В. Ліп’яніна-Гончаренко

підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод ідентифікації ознак тварин на основі глибокого навчання» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 73 сторінки і містить 5 ілюстрацій, 7 таблиць, 2 додатки та 36 використаних джерел.

Метою даної кваліфікаційної роботи є розроблення та експериментальне обґрунтування методу дрібнозернистої класифікації птахів, придатного для використання в умовах обмеженої кількості розмічених даних.

Методи досліджень: аналіз наукової літератури, методи глибокого навчання, самоконтрольоване навчання, метричні методи класифікації, статистичний аналіз, порівняльний аналіз.

Результати дослідження: розроблено гібридний метод, що поєднує самонавчену модель DINOv2 для вилучення ознак та непараметричні класифікатори. Метод дозволив досягти точності до 91% на повному наборі даних та 88% у Few-Shot сценаріях (5 зразків) без необхідності ресурсоємного донавчання. Підтверджено високу ефективність методу при роботі із закритими зображеннями та в умовах OOD-валідації.

Результати роботи можуть успішно застосовуватися в системах екологічного моніторингу, «розумних» фотопастках та мобільних додатках для орнітологів, де важлива автономність та швидкість роботи.

Ключові слова: ДРІБНОЗЕРНИСТА КЛАСИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА, ГЛИБОКЕ НАВЧАННЯ, ПІДВИЩЕННЯ ТОЧНОСТІ КЛАСИФІКАЦІЇ, РОЗПІЗНАВАННЯ ПТАХІВ.

ABSTRACT

Qualification work on the topic "Method for animal feature identification based on deep learning" for the Master's degree in specialty 122 "Computer Science", educational program "Computer Science", consists of 73 pages and contains 5 figures, 7 tables, 2 annexes, and 36 references.

The purpose of this qualification work is to develop and experimentally substantiate a method for fine-grained bird classification suitable for use under conditions of limited labeled data.

Research methods: analysis of scientific literature, deep learning methods, self-supervised learning, metric classification methods, statistical analysis, comparative analysis.

Research results: a hybrid method combining the self-supervised DINOv2 model for feature extraction and non-parametric classifiers has been developed. The method achieved accuracy of up to 91% on the full dataset and 88% in Few-Shot scenarios (5 samples) without the need for resource-intensive fine-tuning. The high efficiency of the method when working with occluded images and under OOD validation conditions has been confirmed.

The results of the work can be successfully applied in ecological monitoring systems, "smart" camera traps, and mobile applications for ornithologists, where autonomy and speed of operation are important.

Keywords: FINE-GRAINED CLASSIFICATION, NEURAL NETWORK, DEEP LEARNING, IMPROVING CLASSIFICATION ACCURACY, BIRD RECOGNITION.

ЗМІСТ

Перелік умовних позначень і скорочень 7
Вступ 8
1 Аналіз предметної області і постановка завдання дослідження 11
1.1 Обґрунтування актуальності дослідження, аналіз стану вирішення задачі 11
1.2 Аналіз відомих рішень 15
1.3 Постановка задачі 22
Висновки до розділу 1 25
2 Результати теоретичних досліджень у межах наукової задачі 26
2.1 Математичні основи методу та архітектура Vision Transformer 26
2.2 Архітектура запропонованого гібридного методу 33
2.3 Непараметрична класифікація у запропонованому методі 34
2.4 Сценарії та метрики оцінювання 36
Висновки до розділу 2 38
3 Практичне використання отриманих результатів 39
3.1 Програмна реалізація та інструментарій дослідження 39
3.2 Результати пропонованого гібридного методу 42
3.3 Аналіз OOD-валідації 46
3.4 Тестування стійкості до перекриття у реальних умовах 47
Висновки до розділу 3 52
Висновки 53
Список використаних джерел 56
Додаток А Основна частина коду для запропонованого гібридного методу 60
Додаток Б Апробація отриманих результатів 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

CNN — згорткова нейронна мережа (Convolutional Neural Network)

ViT — візійний трансформер (Vision Transformer)

DINOv2 — метод самонавченого (self-supervised) навчання для вилучення ознак

SSL — самонавчене навчання (Self-Supervised Learning)

k-NN — метод k найближчих сусідів (k-Nearest Neighbors)

OOD — позарозподільна перевірка (Out-of-Distribution)

FGVC — дрібнозерниста класифікація (Fine-Grained Visual Classification)

LR — швидкість навчання (learning rate)

TP — істинно позитивні класифікації (True Positives)

TN — істинно негативні класифікації (True Negatives)

FP — хибно позитивні класифікації (False Positives)

FN — хибно негативні класифікації (False Negatives)

Accuracy (A) — загальна точність

Precision (P) — точність

Recall (R) — повнота

F1-score (F1) — F1-міра

ВСТУП

Класифікація видів птахів є дуже важливою для екологічних досліджень та захисту біорізноманіття. Вони є ключовою складовою підтримки екологічного балансу через розповсюдження насіння, запилення та контроль шкідників, що є переважаючим для біорізноманіття та здорової екосистеми. Точна ідентифікація та класифікація видів птахів важлива для екологічних досліджень та збереження біорізноманіття, перш за все, для моніторингу змін навколишнього середовища.

Через вирубку лісів, індустріалізацію та глобальне потепління багато видів птахів зникли повністю, є на межі вимирання або значно менших кількостях, що є критичним для певних екосистем [1].

Традиційні методи класифікації на основі морфологічного аналізу обмежені, трудомісткі та схильні до суб'єктивності спостерігача. Однак, досягнення в цифровій візуалізації та техніках розпізнавання зображень високої інтенсивності надають надійні рішення для автоматичної класифікації птахів та роблять можливою ідентифікацію вимираючих видів, їх контроль у природніх середовищах існування.

Дослідження у сфері класифікації птахів в більшості проводяться безпосередньо в середовищах їх існування, зокрема лісах, горах та полях, що сильно віддалені від усієї інфраструктури та де умови дуже обмежені.

Сучасні підходи, що базуються на Згорткових Нейронних Мережах [2] та трансферному навчанні [3], хоча й надзвичайно точні, але часто є обчислювально дорогими, вимагають значного донавчання та складних ансамблевих рішень, що вимагає складного устаткування.

Також, через свою складність такі моделі порівняно гірше вирішують прикладні задачі, коли моделі дають лише декілька прикладів зображень виду і потім вона повинна класифікувати цей вид серед інших зображень - Few Shot експерименти [4].

Мета роботи полягає у розробленні та експериментальному обґрунтуванні гібридного методу дрібнозернистої класифікації птахів на основі самонавчених візуальних ознак (DINOv2/ViT) та непараметричних класифікаторів (k-NN і

прототипів), придатного для використання в умовах обмеженої кількості розмічених даних.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

1) проаналізувати предметну область дрібнозернистої класифікації та наявні підходи до розпізнавання птахів;

2) обґрунтувати вибір базової архітектури (Vision Transformer [5]) і методу самонавчання (DINOv2) для вилучення ознак;

3) сформувати конвеєр підготовки даних і процедури формування ембедингів для навчальної та тестової вибірок;

4) реалізувати непараметричні стратегії класифікації (k-NN [6], прототипи [7]) і їх поєднання у гібридному рішенні;

5) визначити сценарії оцінювання (у т.ч. few-shot та out-of-distribution [8]) і метрики якості, виконати експериментальне порівняння з базовими підходами;

6) оцінити робастність методу до часткового перекриття об’єкта та проаналізувати типові помилки за матрицями плутанини.

Об’єктом дослідження є процес автоматизованої ідентифікації видів птахів за цифровими зображеннями у польових умовах.

Предметом дослідження є методи вилучення візуальних ознак на основі самонавчання та методи непараметричної класифікації/ансамблювання для дрібнозернистого розпізнавання.

Методи дослідження: аналіз і систематизація наукових праць; методи математичного моделювання та оптимізації; експериментальні методи машинного навчання; порівняльний аналіз; статистична інтерпретація метрик якості; аналіз помилок за матрицею плутанини.

Інформаційну базу дослідження становлять відкриті набори зображень птахів (та/або їх підмножини), а також попередньо навчені ваги DINOv2; програмна реалізація виконана мовою Python із використанням бібліотек PyTorch [9], TorchVision [10] та супровідних інструментів.

Практичне значення полягає у створенні програмного модуля, який забезпечує ідентифікацію видів птахів на основі ембедингів без повного

донавчання важких CNN-моделей, що зменшує обчислювальні витрати та спрощує адаптацію до нових умов зйомки.

Наукова новизна дослдіження. Удосконалено метод дрібнозернистої класифікації видів птахів за зображеннями шляхом поєднання самонавчених візуальних ембедингів DINOv2/ViT із непараметричними стратегіями розпізнавання (k-NN та прототипним класифікатором) у гібридному рішенні, що підвищує робастність до доменного зсуву та часткових оклюзій і забезпечує ефективну адаптацію в few-shot сценаріях без ресурсоємного донавчання моделі.

Апробація результатів дослідження. Основні теоретичні положення кваліфікаційної роботи та практичні результати дослідження доповідалися й обговорювалися на VI Всеукраїнській студентській науковій конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» [11], яка відбулася 21 червня 2024 року в м. Рівне, а також на науково-практичній конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR–2025) [12], яка відбулася в місті Тернополі 27–29 травня 2025 року. Копії публікацій розміщені в додатку Б.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Обґрунтування актуальності дослідження, аналіз стану вирішення задачі

Питання класифікації та дослідження птахів на даний момент є актуальною та потрібною темою, адже вони є важливою ланкою екологічних досліджень. Класифікація та облік багатьох виидів птахів набуває актуальності через те, що люди знищують їх природні екологічні зони, змінюється клімат тощо. Через це науковцям важливо контролювати процеси міграції, вести облік осіб певних класів, досліджувати, яку користь ті чи інші птахи приносять і що буде у разі зниження їх популяції.

Це дуже трудомісткий процес, який вимагає багато сил та часу науковців. Такі дослідження часто проводяться в польових умовах, коли результат потрібен тут і зараз, а не за якийсь час, адже іншої можливості побачити конкретного птаха чи клас потрібно буде чекати знову дуже довго. В багатьох випадках, доступ до інтернету чи складного обладнання обмежений, а є необхідність класифікувати птахів на місці.

Традиційні методи моніторингу, що базуються на візуальному спостереженні фахівцями-орнітологами, є обмеженими у масштабуванні. Вони вимагають значних людських ресурсів, високої кваліфікації експертів та значних часових затрат. У цьому контексті автоматизовані системи комп'ютерного зору набувають критичного значення.

Особливістю задачі класифікації птахів є її належність до класу задач розпізнавання дрібнозернистих об'єктів. На відміну від загальної класифікації об'єктів, наприклад, розрізнення кота від автомобіля, вона вимагає розрізнення підкласів одного базового класу. У випадку птахів це означає необхідність ідентифікувати сотні видів, які можуть мати мінімальні візуальні відмінності, наприклад, форма дзьоба, відтінок оперення навколо очей, візерунок крил, при цьому внутрішньокласова варіативність може бути високою через вік птаха, сезонні зміни оперення або статевий диморфізм.

Окрім внутрішньокласової варіативності, задача ускладнюється біологічним явищем мімікрії та природної гібридизації. Деякі види птахів еволюційно розвинули схожість з іншими видами для захисту від хижаків. Для комп'ютерного зору це створює проблему хибно-позитивних спрацювань. Традиційні CNN, які часто покладаються на текстуру оперення, у таких випадках дають збій.

Також існує проблема "частина-ціле". У багатьох випадках птах на зображенні може бути частково перекритий гілками або знаходитися у такій позі, що видно лише голову. Для людини-експерта цього часто достатньо для ідентифікації, оскільки людина використовує семантичне розуміння анатомії. Згорткові мережі, навчені "з учителем", часто сприймають такі зображення як шум, оскільки в навчальній вибірці переважали фотографії птахів у повний зріст. Саме тому підходи, засновані на механізмах уваги, які здатні виділяти семантично значущі регіони незалежно від їх розташування, є більш перспективними.

Метою цієї роботи було розробити метод, що досягатиме високих показників точності розпізнавання тварин в умовах обмежених прикладів та ресурсів, впровадивши сучасніший, легший та гнучкіший підхід, що базується на Self-Supervised Learning. Таким чином, повна система з його використанням зможе працювати автономно в майже будь-яких умовах та, як один із варіантів використання, зможе сповістити науковців у разі класифікації потрібних їм птахів.

Запропоновано гібридний метод машинного навчання, який складається з двох етапів:

1. Навчання та подальше використання потужної SSL-моделі DINOv2 (ViT-L/14) для генерації високоякісних векторів ознак без донавчання.

2. Застосування простих, швидких та непараметричних класифікаторів безпосередньо до цих ембедингів.

Актуальність такого підходу полягає у його винятковій ефективності. Він дозволяє створювати локальні системи розпізнавання без необхідності у хмарних ресурсах чи GPU в цілому, що є критично важливим для польових центрів спостережень.

Використання SSL-ембедингів робить систему гнучкою для розпізнавання будь-яких видів, навіть якщо кількість їх зразків обмежена. Вона повинна

демонструвати високу продуктивність в умовах дуже обмежених даних, що є дуже актуальним для реальних завдань біорізноманіття.

Однією з фундаментальних проблем, яка ускладнює створення автоматизованих систем для екології, є природний незбалансований розподіл даних. У статистиці це явище відоме як розподіл із «довгим хвостом». У будь-якій екосистемі існує невелика кількість домінуючих видів, наприклад, горобці, голуби, ворон), які зустрічаються повсюди і для яких легко зібрати тисячі фотографій. Водночас, переважна більшість видів є рідкісними, ендемічними або знаходяться під загрозою зникнення. Для таких видів кількість доступних візуальних даних може обчислюватися одиницями.

Традиційні методи машинного навчання, що базуються на навчанні з учителем, мають тенденцію ігнорувати класи з «хвоста» розподілу, оптимізуючи загальну точність за рахунок домінуючих класів. Якщо модель навчена на 1000 зображень горобців і 5 зображеннях рідкісного орла, вона, найімовірніше, класифікуватиме орла як горобця, оскільки це статистично вигідніше для мінімізації глобальної помилки.

Саме тому критично важливою є розробка методів, які не залежать від обсягу навчальної вибірки, а базуються на семантичній схожості. Підхід Few-Shot Learning, який розглядається в цій роботі, є прямою відповіддю на виклик «довгого хвоста», дозволяючи системі ефективно працювати з видами, представленими мінімальною кількістю прикладів.

Задача розпізнавання видів птахів відноситься до специфічного класу проблем комп'ютерного зору, відомого як дрібнозерниста візуальна класифікація [13].

На відміну від загальної класифікації об'єктів, де категорії семантично віддалені, FGVC оперує підкатегоріями одного батьківського класу. Це накладає специфічні вимоги до алгоритмів, оскільки візуальні відмінності між класами є мінімальними, а варіативність всередині одного класу може бути значною.

Основною проблемою FGVC є висока внутрішньокласова дисперсія та низька міжкласова відмінність. Внутрішньокласова варіативність зумовлена тим, що птахи одного виду можуть виглядати кардинально по-різному залежно від

декількох факторів: віку, статевого диморфізму, сезонних змін, а також пози об'єкта та умов освітлення.

У той же час, міжкласова подібність є критично високою - різні види, наприклад, родини очеретянок, можуть відрізнятися лише формою махових пір'їн або незначним відтінок надбрівної смуги, що важко розрізнити навіть для людського ока без спеціального обладнання.

Класичні згорткові мережі, які навчаються на функції втрат перехресної ентропії, часто не здатні вловити ці локальні дискримінативні ознаки, фокусуючись на глобальній формі об'єкта, яка у всіх птахів є схожою. Саме тому виникає необхідність у використанні методів, що здатні формувати більш щільні кластери ознак у векторному просторі.

Також варто зазначити еволюцію підходів Self-Supervised Learning [14], які вирішують проблему залежності від розмітки даних. Ранні методи SSL, такі як SimCLR або MoCo, покладалися на контрастивне навчання, де модель вчилася зближувати позитивні пари і віддаляти негативні. Однак ці методи вимагали великих розмірів пакетів та значних обчислювальних ресурсів для підтримання черги негативних прикладів.

Метод DINO, обраний у цій роботі, використовує парадигму дистиляції знань без використання негативних пар, що дозволяє навчати модель фокусуватися на локальних семантичних частинах об'єкта без явного вчителя. Це робить його архітектурно більш придатним для задач FGVC, де локальні деталі мають вирішальне значення.

Окремою, але критично важливою проблемою сучасних досліджень, яку часто ігнорують у гонитві за точністю, є енергоефективність. Концепція "Green AI" наголошує на необхідності врахування вуглецевого сліду при навчанні моделей. Навчання ансамблю з трьох важких CNN, як у згаданих аналогах, вимагає сотень годин роботи потужних GPU, що призводить до значних викидів CO_2.

У контексті екологічного моніторингу це створює парадокс - використовуються екологічно шкідливі методи для захисту екології. Більше того, розгортання систем моніторингу часто відбувається на периферійних пристроях,

таких як звичайні ноутбуки, Raspberry Pi, NVIDIA Jetson або навіть автономних дронах. Ці пристрої мають жорсткі обмеження по живленню та тепловідведенню.

Важкі CNN-ансамблі фізично не можуть бути розгорнуті на такій архітектурі без значної затримки або перегріву пристрою. Тому перехід до методів, що не вимагають обчислень градієнтів на пристрої, є не просто альтернативою, а технічною необхідністю.

1.2 Аналіз відомих рішень

Як відповідь на ці потреби, техніки глибокого навчання відкрили нові виміри автоматизації та точності в детальному розпізнаванні видів птахів. Наприклад, Згорткові Нейронні Мережі були дуже ефективними в обробці великих наборів даних зображень, де є автоматичне або значне зменшення залежності від ручного вилучення ознак. Ця автоматизація дозволяє CNN виявляти тонкі характеристики, такі як візерунки оперення, забарвлення, структура дзьоба або форма тіла, що є критичним для ідентифікації видів.

Коли маркованих даних недостатньо, поширеною практикою стало трансферне навчання. Воно дозволяє адаптувати моделі, попередньо навчені на масивних датасетах, до вузькоспеціалізованої задачі, як-от розпізнавання птахів. Це прискорює навчання та дозволяє досягти високої точності, але така модель залишається складною і все ще більше підходить для розпізнавання «так -це птах», а не конкретного виду птахів.

Прагнучи до максимальної точності, дослідники, наприклад Richard et al., 2025 [15] часто вдаються до складних ансамблів та комплексних класифікаційних голів - наприклад, на основі нечіткої логіки. Цей підхід SOTA досяг вражаючої точності у понад 95%.

Однак цей метод має суттєві практичні обмеження:

1. Донавчання кількох глибоких CNN вимагає потужних ресурів, хмарних обчислень та значного часу.
2. Реалізація складних ансамблевих логік є дуже важкою та потребує ретельного налаштування.

3. Моделі, донавчені на конкретних зразках, досить погано узагальнюють знання на нові, раніше не бачені класи або на ті, що представлені з дуже малою кількістю прикладів.

Нещодавні прориви спираються на моделі глибокого навчання, зокрема архітектури CNN, для покращення якості та ефективності класифікації. Розглянемо останні дослідження у цій сфері:

- Song, H (2024) [16] досліджували покращений ResNet-152, що досяг високих показників точності та F1 – 94% на датасеті Birds525. Переваги архітектури ResNet допомогли вирішити проблему зникаючих градієнтів, хоча, як зазначається, залишились обмеження щодо закритих зображень.
- Cai, R. (2023) [17] використовували EfficientNetB0 з аугментацією та трансферним навчанням, досягнувши 86.7% точності на Birds525, хоча й відзначаються труднощі з видами, що мають тонкі візуальні відмінності.
- Farman et al. (2023) [18] продемонстрували високу точність - 92% з VGG16 на малому наборі (20 видів), але показали значне падіння продуктивності - до 60% при масштабуванні до 525 видів, що підкреслює проблеми масштабованості традиційних CNN.
- Wang et al. (2023) [19] використали ShuffleNetV2 з механізмами уваги, досягши 87.02% на CUB-200-2011, що робить його придатним для мобільних пристроїв, але з компромісом у точності дрібнозернистої класифікації.
- Kondaveeti et al. (2023) [20] з MobileNetV2 досягли 84.5% точності, але відзначили проблеми з даними високої роздільної здатності.
- Інші підходи включали модифікації YOLOv5 [21] для дрібнозернистої класифікації - 92% точності/
- Використання трансформерів [22], які показали 89.4% точності, але були обчислювально інтенсивними.
- Гібридні підходи [23], що поєднали YOLOv5 з EfficientNetB3, досягли високої точності 98%, але не вирішували проблему закритих зображень, випадків із декількома прикладами та були надзвичайно важкими.

Узагальнення основних цих підходів представлене у таблиці 1.1.

Таблиця 1.1 - Якісне порівняння існуючих моделей класифікації

Дослідження	Ключовий внесок	Переваги	Недоліки
Enhancing occluded and standard bird object recognition [15]	Досягнуто великої точності при звичайних завдання розпізнавання птахів - більше 95%	Висока точність при звичайних та закритих зображеннях	Складний, дорогий та довгий процес навчання, високі вимоги до ресурсів та відносно середні результати у сценаріях декількох прикладів
Bird image classification using convolutional neural network and transfer learning [16]	Запропоновано покращену модель ResNet-152 для класифікації птахів, досягнуто високої точності	Покращує класифікацію, вирішуючи проблему зникаючих градієнтів за допомогою залишкових шарів	Має труднощі із закритими зображеннями, що впливає на продуктивність
A convolutional neural network for bird's classification using [17]	Використано EfficientNetB0 з аугментацією даних та transfer learning для класифікації	Досягнуто точності 86.7%, що демонструє ефективність transfer learning	Складно класифікувати певні види через незначні візуальні відмінності
Deep convolutional neural network for automated bird species classification [20]	Види птахів класифіковано за допомогою MobileNetV2 з точністю 84.5%	Досить низькі мінімальні обчислювальні вимоги	Має проблеми з класифікацією даних високої роздільної здатності, де більші моделі перевершують
Effective classification of bird' species based on transfer learning [22]	Запропоновано Hybrid Granularities Transformer для дрібнозернистої класифікації	Покращує складні завдання категоризації шляхом вилучення як локальних, так і глобальних ознак	Обчислювально вимоглива, що ускладнює застосування в реальному часі або з низькими ресурсами
Significant feature suppression and cross-feature fusion networks for fine-grained visual classification [23]	Розроблено SFSCF-Net для FGVC шляхом злиття крос-ознак із придушенням значущості	Покращений фокус на ознаках та надійне семантичне представлення для дрібнозернистої класифікації птахів	Висока обчислювальна складність, що унеможливлює використання в реальному часі або з обмеженими ресурсами

Розглянемо архітектуру моделі у останньому існуючому дослідженні [15], що досягло великої точності при стандартних та закритих зображеннях. Базовий підхід, описаний у цьому дослідженні був багатоетапним та складним. Автори обрали дев'ять різних CNN-архітектур для початкового тестування. Всі моделі пройшли донавчання на навчальному наборі даних з використанням оптимізатора, що було визначено як оптимальна конфігурація в їхньому абляційному дослідженні.

На другому етапі автори досліджували гібридний підхід, де три найкращі CNN використовувались як екстрактори ознак. Вилучені карти ознак конкатенувалися і подавалися на вхід класичних ML-класифікаторів. Цей підхід показав високі результати - біля 95% точності.

Фінальний та найскладніший метод, запропонований в оригінальній статті, - це ансамбль на основі нечіткої логіки. На відміну від простого зваженого усереднення, цей метод динамічно коригує ваги для кожного окремого зображення.

Процес працював наступним чином:

1. Для вхідного зображення кожна з трьох CNN-моделей генерує вектор ймовірностей.
2. Обчислюється "оцінка впевненості" для кожної моделі як максимальна ймовірність у векторі.
3. Ці три оцінки впевненості подаються в Систему Нечіткого Висновку.
4. СНВ, на основі набору правил, обчислює динамічні ваги.
5. Кінцевий прогноз є зваженою сумою прогнозів, використовуючи ці динамічні ваги.

Ключовим аспектом досліджень-аналогів є те, наскільки ретельно авторам доводиться налаштовувати процес донавчання CNN. Таблиця 1.2 демонструє порівняння конфігурацій оптимізаторів у дослідженнях, використовуючих важкі CNN методи. Можна чітко побачити, що вибір конфігурації є критичним. Наприклад, ResNet101V2 з ADAM показує 89.32% точності, тоді як з RMSProp - 96.04%. Це підкреслює чутливість процесу донавчання до гіперпараметрів, що вимагає значних інженерних зусиль для підбору.

Таблиця 1.2 - Порівняння конфігурацій оптимізаторів у дослідженнях, використовуючих важкі CNN методи

Назва моделі	Оптимізатор	LR	Точність при тестовому наборі (%)	Точність при закритих зображеннях (%)
DenseNet121	ADAM	1e-3	90.38	86.54
	ADAM	1e-5	91.41	88.00
	RMSProp	1e-3	91.68	87.87
	RMSProp	1e-5	95.07	91.77
DenseNet169	ADAM	1e-3	87.18	84.85
	ADAM	1e-5	91.20	87.64
	RMSProp	1e-3	90.96	86.67
	RMSProp	1e-5	94.96	92.14
ResNet101V2	ADAM	1e-3	89.32	85.45
	ADAM	1e-5	92.05	87.87
	RMSProp	1e-3	91.82	87.15
	RMSProp	1e-5	96.04	91.41

Таблиця 1.3 показує вплив "заморожування" шарів. Для всіх трьох топ-моделей, продуктивність при 0% тренуваних шарів є найнижчою. Найкращі результати досягаються лише при 100% донавчанні. Це доводить, що для supervised CNN-моделей просте вилучення ознак є недостатнім, і необхідне дороге та ресурсомістке донавчання.

Отже, весь багатоетапний процес оптимізації важких CNN ансамблів: вибір оптимізатора, LR, відсоток заморожених шарів, стратегія злиття, розподіл ваг, демонструє високу складність та інженерні витрати базових методів. Також, такі методи неможливо використовувати в польових умовах, як цього вимагає безліч сучасних досліджень. До того ж, ці високі показники точності лише у звичайних випадках, а при роботі з Few Shot сценаріями, точність падає до 50-70%, бо моделі погано адаптуються та дуже узагальнюють дані відносного того, що вони вже знають.

Таблиця 1.3 - Порівняння продуктивності тренуваних та заморожених шарів

Назва моделі	Шари, що навчаються (%)	Стандартні зображення (%)	Закриті зображення птахів (%)
DenseNet121	100%	95.07	91.77
	50%	91.68	87.87
	0% (Заморожені)	90.38	86.54
DenseNet169	100%	94.96	92.14
	50%	90.96	86.67
	0% (Заморожені)	87.18	84.85
ResNet101V2	100%	96.04	91.41
	50%	91.82	87.15
	0% (Заморожені)	89.32	85.45

Таблиця 1.4 узагальнює ключові відмінності між оригінальним дослідженням Richard et al. [15] та запропонованим гібридним методом, що є метою мого дослідження.

Таблиця 1.4 - Порівняння підходів

Напрям	Оригінальне дослідження [15]	Запропонований гібридний SSL-метод
Архітектура	CNN-моделі (ResNet, DenseNet, InceptionV3) з TL	DINOv2 (ViT-L/14) self-supervised вектори
Класифікаційна частина (head)	Повнозв'язний шар (донавчання)	k-NN класифікатор + прототипи класів
Навчання	Transfer learning з fine-tuning	Без навчання — frozen embeddings
Датасети	CUB_200_2011, Birds525	Ті самі (повністю локально)
OOD-валідація	External subset	External + Cross-dataset (CUB ↔ Birds525)
Оптимізація	GPU cloud	Оптимізація під середньостатистичні пристрої

Оглянуті дослідження зосереджені на керованому трансферному навчанні, цей підхід має фундаментальне обмеження: він покладається на якість міток ImageNet [24]. Моделі "вчаться" розрізняти 1000 класів ImageNet, і це знання потім переноситься на птахів.

Парадигма Self-Supervised Learning пропонує інший шлях - модель навчається виявляти візуальні ознаки, не бачачи жодних людських міток. Моделі, такі як DINOv2, навчаються за принципом "узгодженості" - різні аугментовані версії одного і того ж зображення повинні давати схожі вектори ознак, тоді як вектори різних зображень мають бути відмінними.

DINOv2, зокрема, використовує архітектуру Vision Transformer [5] і був навчений на масивному, некуратованому наборі даних LVD-142M. В результаті DINOv2 вивчає надзвичайно робастні та узагальнені ознаки, які кодують не лише клас об'єкта, але і його семантичну структуру, наприклад: "голова", "крило", "дзьоб", навіть не знаючи цих слів.

Ознаки, вивчені DINOv2, є настільки багатими, що вони усувають необхідність у складному донавчанні. Гібридний метод, що використовує його у поєднанні з простими векторними методами як k-nn забезпечує дивуючі результати в порівнянні з існуючими аналогами.

Це дозволяє замінити складний, повільний і "важкий" ансамбль CNN, забезпечуючи наступні покращення:

1. Досягнення конкурентної, а той вищої точності, витративши значно менше часу та обчислювальних ресурсів.
2. SSL-ознаки точніші, ніж TL, коли доступна лише мала кількість зразків (1-5) на клас.
3. SSL-ознаки краще переносяться між різними даними, ніж спеціалізовані донавчені моделі.

Для забезпечення прямого порівняння з базовими дослідженнями, використовуються ті самі два основні набори даних:

1. Caltech-UCSD Birds-200-2011 [25] - класичний датасет для дрібнозернистої класифікації. Використовується 5 994 навчальних зображень та 5 794 тестових зображень, що охоплюють 200 видів птахів.

2. Birds525 Species-Image Classification [26] - більший та різноманітніший набір даних. Використовується 84 635 навчальних зображень, 2 625 тестових та 2 625 валідаційних зображень, що охоплюють 525 видів.

1.3 Постановка задачі

Актуальність теми зумовлена стрімким зростанням обсягів візуальних даних у біологічному моніторингу, природоохоронних проєктах і громадянській науці, де все частіше використовуються фотоспостереження, польові знімки з мобільних пристроїв і матеріали з фотопасток. Для таких застосувань критично важливо автоматизувати ідентифікацію видів птахів, оскільки ручне маркування потребує високої кваліфікації експерта-орнітолога, є часовитратним та погано масштабується при збільшенні наборів даних. Водночас точність розпізнавання безпосередньо впливає на коректність оцінювання біорізноманіття, виявлення змін у популяціях та оперативність ухвалення рішень у сфері охорони природи, що робить задачу дрібнозернистої класифікації птахів суспільно значущою та практично затребуваною.

Дрібнозерниста класифікація (Fine-Grained Visual Classification, FGVC) є однією з найскладніших задач комп’ютерного зору, оскільки класи часто відрізняються мінімальними візуальними ознаками (тонкі відмінності оперення, пропорцій тіла, форми дзьоба чи забарвлення), тоді як варіативність усередині одного класу може бути значною через вік, стать, сезонну зміну оперення, позу, освітлення та фон. У реальних польових умовах додатковою проблемою є часткове перекриття об’єкта (гілками, листям, іншими птахами), розмиття, низька роздільна здатність та нетипові ракурси. Це призводить до того, що класичні підходи, ефективні для “грубих” категорій, демонструють нестабільність на FGVC та потребують спеціально підібраних представлень ознак і процедур оцінювання.

Традиційні глибокі згорткові мережі (CNN), які домінували в задачах розпізнавання зображень, забезпечують високу якість за наявності великих обсягів розмічених даних, однак у дрібнозернистих задачах їх застосування часто ускладнене дефіцитом розмітки та високою ціною її отримання. Для конкретних

видів птахів створення репрезентативних, збалансованих і якісно розмічених вибірок є ресурсомістким і потребує участі фахівців, а отже обмежує можливість швидкого масштабування системи на нові території, сезони чи умови зйомки. Крім того, повне донавчання великих CNN-моделей для кожної нової підзадачі підвищує обчислювальні витрати і ускладнює впровадження в прикладні системи з обмеженими ресурсами.

У цьому контексті особливої актуальності набувають сучасні трансформерні архітектури для комп'ютерного зору (Vision Transformer, ViT) та підходи самонавченого навчання (self-supervised learning), зокрема DINOv2, які дозволяють формувати універсальні візуальні репрезентації без необхідності повної супервізованої адаптації на конкретній предметній області. Використання самонавчених ознак дає змогу переносити знання, здобуті на великих не розмічених або слабо розмічених наборах, у вузькі задачі, де якісна розмітка обмежена. Для дрібнозернистої класифікації це означає можливість отримання інформативних ембедингів, чутливих до тонких відмінностей між видами, та створення моделей, які швидше адаптуються до нових класів і доменів.

Окремим чинником актуальності є практична потреба у методах, здатних працювати в режимах few-shot та out-of-distribution (OOD), коли система має розпізнавати класи при дуже малій кількості прикладів або в умовах доменного зсуву (інша камера, інші погодні умови, інший фон, інший регіон). Саме такі сценарії є типовими для польових досліджень і прикладних сервісів, де неможливо гарантувати сталість розподілу даних. Тому поєднання самонавчених ембедингів із непараметричними методами класифікації (k-NN, прототипні підходи) є актуальним напрямом, оскільки забезпечує швидке "навчання" на нових класах без тривалого донавчання параметрів і дозволяє гнучко керувати компромісом між точністю та обчислювальною складністю.

Нарешті, актуальність роботи підсилюється необхідністю підвищення робастності систем розпізнавання до часткових оклюзій і типових спотворень реальних зображень, а також потребою в зрозумілому аналізі помилок для подальшого вдосконалення моделей. Застосування матриць плутанини та порівняння стратегій класифікації на спільному просторі ознак дозволяє виявляти

проблемні групи видів, оцінювати причини хибних класифікацій і формувати практичні рекомендації щодо покращення набору даних та алгоритмів. Таким чином, дослідження гібридного підходу на основі DINOv2/ViT та непараметричних класифікаторів є своєчасним і обґрунтованим як з наукової, так і з прикладної точки зору, оскільки спрямоване на створення ефективної, адаптивної та ресурсозберігаючої системи дрібнозернистого розпізнавання птахів.

Мета роботи полягає у розробленні та експериментальному обґрунтуванні гібридного методу дрібнозернистої класифікації птахів на основі самонавчених візуальних ознак (DINOv2/ViT) та непараметричних класифікаторів (k-NN і прототипів), придатного для використання в умовах обмеженої кількості розмічених даних.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

1) проаналізувати предметну область дрібнозернистої класифікації та наявні підходи до розпізнавання птахів;

2) обґрунтувати вибір базової архітектури (Vision Transformer) і методу самонавчання (DINOv2) для вилучення ознак;

3) сформувати конвеєр підготовки даних і процедури формування ембедингів для навчальної та тестової вибірок;

4) реалізувати непараметричні стратегії класифікації (k-NN, прототипи) і їх поєднання у гібридному рішенні;

5) визначити сценарії оцінювання (у т.ч. few-shot та out-of-distribution) і метрики якості, виконати експериментальне порівняння з базовими підходами;

6) оцінити робастність методу до часткового перекриття об’єкта та проаналізувати типові помилки за матрицями плутанини.

Отже, проведений аналіз предметної області підтвердив, що традиційні методи глибокого навчання, попри високу точність, мають критичні обмеження для польового моніторингу через високу ресурсоємність та залежність від великих розмічених вибірок.

Виявлена проблематика дрібнозернистої класифікації та ефект «довгого хвоста» у розподілі біологічних даних обґрунтовують необхідність переходу від навчання з учителем до методів самонавчання. Обраний підхід на базі архітектури

Vision Transformer у поєднанні з непараметричними класифікаторами дозволяє вирішити суперечність між точністю розпізнавання та обчислювальною ефективністю, що робить його перспективним для створення адаптивних систем екологічного моніторингу.

Висновки до розділу 1

1. Проведено системний аналіз предметної області екологічного моніторингу та класифікації птахів. Встановлено, що задача належить до класу Fine-Grained Visual Classification (FGVC), специфіка якої полягає у високій внутрішньокласовій варіативності та мінімальних міжкласових відмінностях. Визначено критичні проблеми існуючих підходів: незбалансованість даних (проблема «довгого хвоста»), наявність оклюзій та необхідність роботи в польових умовах з обмеженими ресурсами.

2. Виконано критичний огляд існуючих методів розпізнавання (CNN, ансамблеві моделі, трансферне навчання). Виявлено, що попри високу точність (понад 95% на стандартних датасетах), традиційні підходи «з учителем» мають суттєві обмеження: надмірна обчислювальна складність, високе енергоспоживання (що суперечить концепції Green AI), низька здатність до узагальнення на рідкісні види (Few-Shot сценарії) та неможливість розгортання на периферійних пристроях.

3. Обґрунтовано доцільність використання гібридного підходу, що базується на поєднанні Self-Supervised Learning моделі DINOv2 (архітектура ViT) та непараметричних класифікаторів (k-NN, прототипи). Показано, що використання семантично багатих SSL-ембедингів дозволяє уникнути ресурсоємного донавчання та забезпечує високу точність класифікації навіть за умови мінімальної кількості прикладів.

4. Сформульовано мету та задачі дослідження, які полягають у розробці енергоефективного методу дрібнозернистої класифікації, здатного працювати в режимах Few-Shot та Out-of-Distribution. Це дозволить вирішити суперечність між необхідністю високої точності розпізнавання біорізноманіття та обмеженнями апаратних ресурсів автономних систем моніторингу.

2 РЕЗУЛЬТАТИ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ У МЕЖАХ НАУКОВОЇ ЗАДАЧІ

2.1 Математичні основи методу та архітектура Vision Transformer

В основі запропонованого гібридного методу лежить використання архітектури Vision Transformer, принципи роботи якої фундаментально відрізняються від класичних згорткових нейронних мереж. Якщо традиційні CNN обробляють зображення локально через ковзне вікно фільтрів, поступово збільшуючи рецептивне поле, то ViT сприймає вхідне зображення глобально, трансформуючи його у послідовність даних.

Формально, вхідне зображення x розглядається як тривимірний масив, розмірність якого залежить від висоти H, ширини W та кількості кольорових каналів C і описується формулою:

$$x \in R^{H \times W \times C} \quad , \tag{2.1}$$

де x - вхідне зображення;

H - висота зображення у пікселях;

W - ширина зображення у пікселях;

C - кількість каналів (зазвичай C=3 для RGB зображень).

На першому етапі обробки це зображення розбивається на сітку фіксованих фрагментів і перетворюється у послідовність плоских патчів. Розмірність кожного такого розгорнутого вектора-патча визначається квадратом його сторони P та глибиною каналів, що представлено формулою:

$$x_p \in R^{N \times (P^2 \cdot C)} \quad , \tag{2.2}$$

де x_p - послідовність розгорнутих патчів;

N - загальна кількість патчів;

P - розмір сторони одного патча (у пікселях);

C - кількість каналів зображення.

Ефективна довжина вхідної послідовності для Трансформера, тобто загальна кількість патчів N, прямо залежить від роздільної здатності вхідного зображення та обраного розміру патча і розраховується за формулою :

$$N = HW/P^2 \ , \tag{2.3}$$

де N - кількість патчів (довжина послідовності);

H, W - геометричні розміри вхідного зображення;

P - розмір сторони патча.

У використовуваній моделі DINOv2 розмір патча становить 14 на 14 пікселів. Кожен такий патч проходить через шар лінійної проекції, відображаючись у латентний векторний простір фіксованої розмірності D. Оскільки операція розбиття на послідовність руйнує просторову структуру, до отриманих векторів додаються спеціальні позиційні ембедінги, що дозволяє моделі «розуміти», де саме знаходився той чи інший фрагмент на оригінальному зображенні.

Ключовим елементом, що забезпечує високу роздільну здатність ознак, є механізм багатоголової самоуваги. На відміну від згортки, яка аналізує лише сусідні пікселі, цей механізм дозволяє кожному патчу бачити всі інші патчі одночасно, моделюючи глобальні семантичні зв'язки. Математично вага уваги, яка визначає важливість одного фрагмента відносно іншого, обчислюється за формулою:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \ , \tag{2.4}$$

де Q (Queries) - матриця запитів;

K (Keys) - матриця ключів;

V (Values) - матриця значень;

d_k - розмірність векторів ключів (використовується для масштабування, щоб уникнути занадто малих градієнтів).

Тут Q, K та V - це матриці, отримані лінійною проекцією вхідних векторів, а ділення на корінь з d_k забезпечує стабільність градієнтів при навчанні. Цей

механізм дозволяє моделі динамічно фокусуватися на дискримінативних частинах об'єкта, наприклад, специфічній формі дзьоба, ігноруючи візуальний шум фону.

Процес вилучення ознак у розробленій системі не передбачає оновлення ваг. Позначимо функцію відображення нейромережі як f з індексом тета, тоді отримання сирого вектора ознак описується формулою :

$$z = f_\theta(I) \quad , \tag{2.5}$$

де z - вихідний вектор ознак (ембединг);

f_θ - функція нейронної мережі з фіксованими параметрами θ;

I - вхідне зображення.

Критично важливим етапом є L2-нормалізація векторів, необхідна для коректної роботи метричних класифікаторів. Нормалізований вектор z обчислюється шляхом ділення вихідного вектора на його Евклідову норму, як показано у формулі:

$$\hat{z} = \frac{z}{\left||z|\right|_2} = \frac{z}{\sqrt{\sum_{i=1}^{D} z_i^2}} \quad , \tag{2.6}$$

де $\hat{z}$ - нормалізований вектор ознак;

z - вихідний ("сирий") вектор;

$\left||z|\right|_2$ - Евклідова норма (L2-норма) вектора;

D - розмірність простору ознак (для ViT-L/14 D=1024);

z_i - i-та компонента вектора.

Для класифікації тестового вектора використовуємо метрику косинусної відстані. У загальному вигляді вона визначається формулою:

$$d(\widehat{z_{test}}, \widehat{z_\iota}) = 1 - \frac{\widehat{z_{test}} \cdot \widehat{z_\iota}}{\left||\widehat{z_{test}}|\right| \cdot \left||\widehat{z_\iota}|\right|} \tag{2.7}$$

де d - косинусна відстань між векторами;

$\widehat{z_{test}}$ - вектор ознак тестового зображення;

$\widehat{z_i}$ - вектор ознак і-го зображення з навчальної вибірки;

Оскільки у запропонованому методі всі вектори попередньо нормалізовані, формула спрощується до вигляду:

$$d(\widehat{z_{test}}, \widehat{z_i}) = 1 - \widehat{z_{test}^T}\widehat{z_i} \quad , \tag{2.8}$$

де $\widehat{z_{test}^T}$ - транспонований вектор тестового зображення;

$\widehat{z_i}$ - вектор з бази знань.

На основі обчислених відстаней застосовується метод k-найближчих сусідів (k-NN). Рішення про приналежність до класу приймається шляхом мажоритарного голосування, яке формалізується формулою:

$$\hat{y} = \arg\max_{c \in C} \sum_{i \in N_k} I(y_i = c) \quad , \tag{2.9}$$

де $\hat{y}$ - передбачений клас для тестового зразка;

C - множина всіх можливих класів;

N_k - множина індексів k найближчих сусідів;

y_i - мітка класу і-го сусіда;

I - індикаторна функція.

Альтернативний підхід використовує прототипи класів. Прототип p з індексом c обчислюється як центроїд усіх навчальних прикладів даного класу згідно з формулою :

$$p_c = \frac{1}{|S_c|} \sum_{(\widehat{z_i}, y_i) \in S_c} \widehat{z_i} \quad , \tag{2.10}$$

де p_c - вектор-прототип (центроїд) класу c;

S_c - множина пар (вектор, мітка) для всіх навчальних прикладів класу c;

$|S_c|$ - кількість елементів у множині;

z_i - нормалізований ембединг і-го зображення.

Для покращення якості ембедингів застосовується конвеєр аугментації. Формально процес отримання аугментованого зображення x з тильдою шляхом застосування трансформації T із множини доступних перетворень T записується як формула :

$$\tilde{x} = T(x), \quad T \sim \mathcal{T} \ , \tag{2.11}$$

де $\tilde{x}$ - аугментоване зображення;

x - вихідне зображення;

T - функція трансформації;

$\mathcal{T}$ - множина допустимих трансформацій.

Використання аугментацій дозволяє мінімізувати емпіричний ризик класифікатора, що розраховується як середнє значення функції втрат на аугментованій вибірці згідно з формулоюЙ:

$$R_{emp}(f) = \frac{1}{M}\sum_{i=1}^{M} L(f(\tilde{x_i}), y_i) \ , \tag{2.12}$$

де R_{emp} - емпіричний ризик;

M - розмір вибірки;

f - функція передбачення класифікатора;

L - функція втрат;

y_i - істинна мітка класу.

Висока якість ембедингів, що використовуються у цій роботі, забезпечується специфічним процесом попереднього навчання моделі DINOv2. На відміну від класичного навчання з учителем, де мінімізується похибка між передбаченням і міткою, DINOv2 використовує підхід самодистиляції знань. Архітектура складається з двох мереж з однаковою структурою, але різними наборами параметрів: мережі-учня g з параметрами theta s та мережі-вчителя g з параметрами theta t.

Вхідне зображення x піддається різним стохастичним перетворенням, утворюючи набір виглядів V. Мережа-учень отримує на вхід глобальні та локальні

вирізки зображення, тоді як вчитель отримує лише глобальні. Задача учня - передбачити вихід вчителя.

Вихідні дані обох мереж нормалізуються за допомогою функції Softmax із параметром температури tau, що дозволяє регулювати різкість розподілу ймовірностей. Ймовірність приналежності до k-го виміру вихідного простору описується формулою :

$$P(x)^{(k)} = \frac{\exp\left(g_\theta(x)^{(k)}/\tau\right)}{\sum_{j=1}^{K} \exp(g_\theta(x)^{(j)}/\tau)} \quad , \tag{2.13}$$

де $P(x)^{(k)}$ - ймовірність приналежності до k-го виміру;

$g_\theta(x)$ - вихідний вектор (логіти);

τ - температурний параметр;

K - загальна кількість вимірів.

Навчання зводиться до мінімізації функції втрат перехресної ентропії між розподілами ймовірностей вчителя та учня. Важливою деталлю є те, що параметри вчителя не оновлюються градієнтним спуском, а формуються як експоненційне ковзне середнє параметрів учня.

Це забезпечує стабільність навчання і запобігає колапсу рішень - ситуації, коли мережа видає однаковий вектор для будь-якого входу. Функція втрат для одного зображення має вигляд:

$$L = \sum_{x_g \in V_{global}} \sum_{x_v \in V} - P_t\left(x_g\right) \log P_s\left(x_v\right) \ , \tag{2.14}$$

де L - значення функції втрат;

V_{global} - множина глобальних кропів зображення;

V - повна множина кропів;

P_t - розподіл ймовірностей мережі-вчителя;

P_s - розподіл ймовірностей мережі-учнем.

Теоретичною основою ефективності запропонованого підходу є "Гіпотеза многовидів". Вона стверджує, що реальні дані високої розмірності, наприклад,

зображення 224x224 пікселів насправді лежать на многовидах значно меншої розмірності, вкладених у цей простір. Завданням нейронної мережі є розгортання цього многовиду таким чином, щоб класи стали лінійно роздільними.

Традиційні згорткові мережі намагаються апроксимувати цю функцію через ієрархію локальних ознак. Однак, моделі DINOv2, завдяки навчанню без вчителя, формують топологію простору ознак, яка базується на семантичній подібності, а не на візуальній схожості пікселів. Це призводить до того, що вектори зображень одного класу групуються у компактні сферичні кластери.

Саме сферичність кластерів є ключовою. При використанні Softmax-шару в класичному навчанні, класи "розтягуються" у конуси, що виходять з початку координат. У Self-Supervised навчанні DINOv2, завдяки нормалізації вихідних даних на одиничну сферу під час тренування, структура простору стає більш рівномірною (Uniformity). Це пояснює, чому прості метричні методи (як k-NN) працюють настільки ефективно: Евклідова відстань на поверхні гіперсфери коректно відображає семантичну відмінність між об'єктами.

Важливою теоретичною проблемою архітектури Трансформерів є їхня інваріантність до перестановок. Оскільки механізм уваги обчислює попарні відношення між усіма патчами одночасно, він не має вбудованого розуміння того, що патч "дзьоб" повинен знаходитися вище патча "шия". Без додаткової інформації перемішування патчів на вході не змінило б вихідний набір векторів, що є неприпустимим для задач комп'ютерного зору.

Для вирішення цієї проблеми у DINOv2 до кожного вхідного вектора патча додається позиційний вектор. Використовується синусоїдальне позиційне кодування, яке дозволяє моделі узагальнювати дані на зображеннях різної роздільної здатності. Для позиції pos та виміру i значення позиційного вектора обчислюється за системою формул:

$$\begin{Bmatrix} PE_{(pos,2i)} = \sin\left(pos/10000^{(2i/d_{(model)})}\right) \\ PE_{\ (pos,2i+1)} = cos(pos/10000^{(2i/d\_(model))}) \end{Bmatrix} \quad , \qquad (2.15)$$

де pos - позиція патча у послідовності;

i - індекс виміру вектора;

$d_{(model)}$ - розмірність векторного простору моделі (для DINOv2 ViT-L/14 це значення дорівнює 1024).

Такі вектори мають унікальні властивості: для будь-якого фіксованого зсуву k, вектор на позиції pos+k може бути представлений як лінійна функція від вектора на позиції pos. Це дозволяє моделі легко навчатися відносним позиціям частин тіла птаха, формуючи структурне розуміння об'єкта.

2.2 Архітектура запропонованого гібридного методу

Запропонований метод кардинально спрощує архітектуру. Донавчений ансамбль CNN та складна система нечіткого висновку замінюється на гібридний метод "екстрактор + класифікатор", що складається з двох етапів.

Зображення один раз проходить через "заморожену" модель DINOv2 для отримання вектора ознак. Класифікація відбувається шляхом порівняння цього вектора з кешованими векторами з тренувального набору або з усередненими прототипами класів.

Запропоновану архітектуру можна побачити на рисунку 2.1.

Усі параметри моделі встановлюються в режим requires_grad = False і модель переводиться в режим model.eval(). Це гарантує, що ваги моделі не оновлюються, і вона використовується виключно як екстрактор ознак.

1. Кожне зображення з навчального та тестового наборів даних проходить через стандартні трансформації (resize 224x224, normalize) і подається на вхід model.forward_features().
2. Вилучається глобальний вектор ознак розмірністю 1024.
3. Вектори L2 - нормалізуються (F.normalize) і зберігаються на диску у вигляді .npy файлів. Цей процес є одноразовим і, завдяки ефективному кешуванню, не повторюється при наступних запусках експериментів.

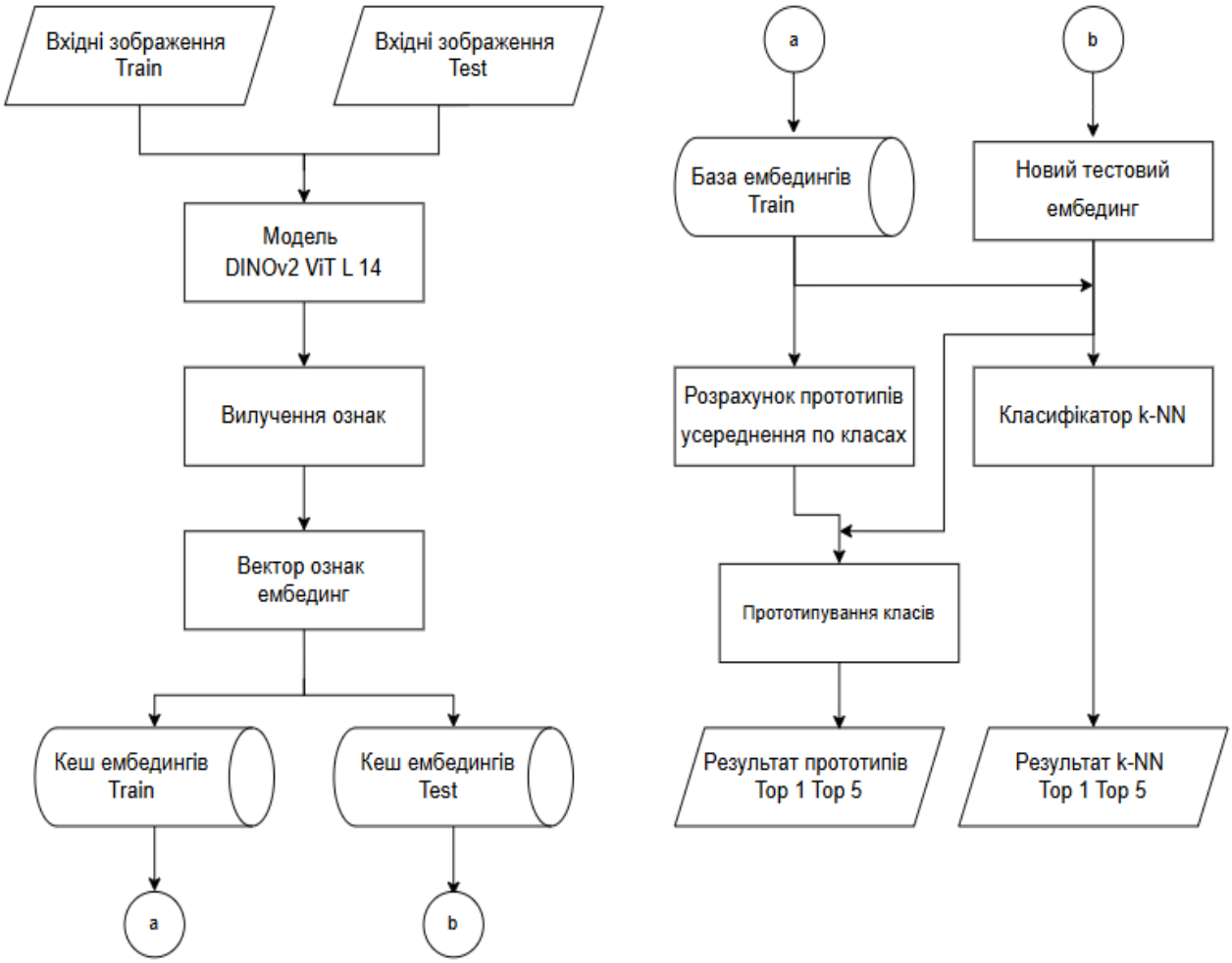

Рисунок 2.1 - Архітектура запропонованого гібридного методу

2.3 Непараметрична класифікація у запропонованому методі

Замість навчання складної класифікаційної "голови" або ансамблю, використовується два метричні підходи:

1. Для тестового ембедингу ми знаходимо k (використано k = 5) найближчих ембедингів з навчального набору даних, використовуючи косинусну відстань, оскільки вектори нормалізовані. Мітка класу визначається мажоритарним голосуванням серед цих k сусідів.

2. Для кожного класу в навчальному наборі ми обчислюємо "прототип" - усереднений вектор усіх ембедингів, що належать цьому класу. Класифікація тестового ембедингу відбувається шляхом знаходження найближчого прототипу за косинусною подібністю.

Важливо обґрунтувати вибір косинусної відстані замість класичної Евклідової метрики L2. У просторах високої розмірності проявляється ефект, відомий як «прокляття розмірності».

Зі збільшенням розмірності простору його об'єм зростає настільки швидко, що доступні дані стають розрідженими. При цьому Евклідова відстань втрачає свою дискримінативну здатність, оскільки відстані між будь-якою парою точок стають майже однаковими.

Зв'язок між квадратом Евклідової відстані та косинусною подібністю для векторів u та v виражається через їх скалярний добуток. Якщо вектори не нормалізовані, цей зв'язок є складним.

Однак у запропонованому методі застосовується попередня L2-нормалізація, що призводить усі вектори до одиничної довжини. У такому випадку квадрат Евклідової відстані лінійно залежить від косинусної відстані, що описується формулою:

$$\left|\left|u - v\right|\right|_2^2 = 2(1 - \cos(u, v)) \quad , \tag{2.16}$$

де $\left|\left|u - v\right|\right|_2^2$ - квадрат Евклідової відстані;

$\cos(u, v)$ - косинусна подібність між векторами;

1 та 2 - математичні константи.

Окрім "прокляття розмірності", при використанні методу k-NN у просторах високої розмірності виникає феномен, відомий як Наявність хабів. Це явище полягає в тому, що певні вектори стають найближчими сусідами для непропорційно великої кількості інших точок даних, незалежно від їхнього класу. Це може призводити до систематичних помилок класифікації.

Теоретичні дослідження показують, що ймовірність появи хабів зростає зі збільшенням розмірності простору. Однак, застосована у роботі L2-нормалізація та використання косинусної відстані частково мітигують цей ефект.

Нормалізація проектує всі точки на поверхню гіперсфери, що зменшує варіацію норм векторів, яка є однією з причин виникнення хабів. Крім того, прототипний метод є менш чутливим до феномену хабів, ніж класичний k-NN,

оскільки усереднення векторів класу згладжує вплив окремих викидів, які могли б діяти як помилкові хаби.

У контексті запропонованого гібридного методу варто розглянути класичну дилему машинного навчання - компроміс зміщення та дисперсії.

Метод k-NN є методом з низьким зміщенням, але високою дисперсією. Це означає, що він може будувати дуже складні межі рішень, які ідеально підлаштовуються під навчальні дані, але можуть бути чутливими до шуму.

Метод прототипів, навпаки, має вище зміщення, оскільки апроксимує клас однією точкою, але значно меншу дисперсію. Він є більш робастим до викидів та більш стабільним.

Запропонований у роботі підхід використання обох методів через зважений ансамбль дозволяє знайти баланс. У сценаріях Few-Shot, де дисперсія k-NN може бути критичною через малу кількість точок, стабільність прототипів дозволяє вирівняти передбачення.

2.4 Сценарії та метрики оцінювання

Для всебічної перевірки ефективності запропонованого методу розроблено систему тестування, що охоплює чотири ключові сценарії, які імітують реальні умови експлуатації.

Перший сценарій - стандартна класифікація: оцінка точності на повному тестовому наборі даних, де класи у навчальній та тестовій вибірках збігаються. Це дозволяє встановити верхню межу можливостей методу.

Другий сценарій - навчання на малій вибірці: тестування в режимі N-way K-shot, де кількість доступних прикладів для кожного класу K варіюється. Цей сценарій є критичним для оцінки адаптивності системи до рідкісних видів.

Третій сценарій - валідація поза розподілом: перевірка здатності моделі узагальнювати знання шляхом навчання на одному датасеті та тестування на іншому.

Четвертий сценарій - стійкість до оклюзій: оцінка падіння точності при штучному перекритті частини зображення від 10% до 50%, що симулює знаходження птаха у густому листі.

Точність є основним показником, що розраховується як відношення правильно класифікованих зразків до загальної кількості зразків. Метрики, що дозволяють всебічно оцінити продуктивність класифікаційних моделей, визначаються формулами :

$$A = \frac{TP + TN}{TP + TN + FP + FN} , \quad (2.17)$$

$$P = \frac{TP}{TP + FP} , \quad (2.18)$$

$$R = \frac{TP}{TP + FN} , \quad (2.19)$$

$$FS = 2 \times \frac{P \times R}{P + R} , \quad (2.20)$$

де A (Accuracy) – загальна точність моделі

P (Precision) – точність позитивного передбачення

R (Recall) – повнота

FS (F1-Score) – гармонійне середнє між Precision і Recall

TP (True Positives) - кількість правильно класифікованих позитивних зразків

TN (True Negatives) - кількість правильно класифікованих негативних зразків

FP (False Positives) - кількість негативних зразків, помилково класифікованих як позитивні

FN (False Negatives) - кількість позитивних зразків, помилково класифікованих як негативні.

Отже, проведені теоретичні дослідження підтверджують доцільність побудови гібридного методу на базі архітектури Vision Transformer. Математична формалізація механізму самоуваги та метричних класифікаторів демонструє, що для аналізу нормалізованих векторів косинусна відстань є не лише ефективною метрикою подібності, але й забезпечує суттєву обчислювальну перевагу. Таким

чином, обґрунтована архітектура, яка передбачає відмову від ресурсомісткого градієнтного навчання на користь одноразового вилучення та кешування ознак, створює необхідне підґрунтя для значного зниження навантаження на апаратну частину при збереженні високої роздільної здатності ознак, критично важливої для дрібнозернистої класифікації птахів.

Висновки до розділу 2

1. Теоретично обґрунтовано вибір архітектури Vision Transformer (ViT) як базової для задачі дрібнозернистої класифікації. Доведено, що механізм глобальної самоуваги (Self-Attention), на відміну від локальних згорток CNN, дозволяє ефективніше моделювати семантичні зв'язки між віддаленими частинами об'єкта, що є критичним для ідентифікації видів птахів за дрібними деталями.
2. Формалізовано математичну модель метричної класифікації у просторі ознак високої розмірності. Показано, що для L2-нормалізованих векторів косинусна відстань є еквівалентом Евклідової метрики, але забезпечує лінійну обчислювальну складність та мітигує ефект «прокляття розмірності», забезпечуючи формування компактних сферичних кластерів класів.
3. Розроблено ресурсоефективну архітектуру гібридного методу, яка базується на використанні «замороженої» моделі DINOv2. Відмова від ресурсомісткого градієнтного навчання (backpropagation) на користь одноразового вилучення та кешування ознак дозволила кардинально знизити вимоги до обчислювальних потужностей, роблячи систему придатною для використання на периферійних пристроях.
4. Запропоновано комбіновану стратегію непараметричної класифікації, що поєднує метод k-найближчих сусідів (k-NN) та метод прототипів. Це дозволило знайти оптимальний баланс між зміщенням та дисперсією (Bias-Variance Tradeoff), забезпечуючи високу адаптивність у сценаріях Few-Shot (завдяки прототипам) та точність при достатній кількості даних (завдяки k-NN).
5. Визначено систему метрик та сценаріїв оцінювання, що включають тестування на стійкість до оклюзій (часткового перекриття) та валідацію поза розподілом (OOD). Це створює необхідне підґрунтя для об'єктивної оцінки робастності розробленого методу в реальних умовах екологічного моніторингу.

3 ПРАКТИЧНЕ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Програмна реалізація та інструментарій дослідження

Програмний комплекс розроблено мовою Python 3.10. У якості інтегрованого середовища розробки обрано Jupyter Lab, що дозволило реалізувати підхід інтерактивних обчислень. Код розділено на логічні блоки, що дає можливість поетапно виконувати завантаження даних, генерацію ознак та експерименти без необхідності перезавантаження нейромережі в оперативну пам'ять при кожній зміні параметрів класифікатора.

Архітектура програмного рішення базується на таких ключових бібліотеках:

1. PyTorch та TorchVision - основний фреймворк для роботи з тензорами та завантаження попередньо навченої моделі. Використано механізм torch.hub для автоматичного завантаження ваг моделі DINOv2.
2. Timm - використано як допоміжну бібліотеку для стандартизації архітектур комп'ютерного зору та попередньої обробки зображень.
3. Scikit-learn - застосовано для розрахунку метрик якості та реалізації базових алгоритмів k-NN для порівняння.
4. Fcmeans - спеціалізована бібліотека, інтегрована для реалізації логіки нечіткої кластеризації у складі розробленого гібридного ансамблю.
5. NumPy та Pandas - для маніпуляцій з масивами ембедингів та структурування результатів експериментів у табличному вигляді.

Важливою особливістю реалізації є відмова від сторонніх індексів пошуку на користь прямих тензорних обчислень засобами PyTorch.

Реалізовано функцію batched_cdist, яка обчислює матрицю попарних Евклідових відстаней між векторами запитів та опорною множиною.

Обчислення виконуються на GPU пакетами, що дозволяє уникнути переповнення відеопам'яті навіть при роботі з великими вибірками, зберігаючи при цьому високу швидкість операцій. Є можливість адаптації коду під більшість пристроїв, навіть тих, що не мають відеоядер взагалі.

Для забезпечення повної відтворюваності експериментів програмний код включає механізми суворої фіксації генераторів псевдовипадкових чисел. На

початку виконання скрипту встановлюються ідентичні значення зерен для бібліотек Numpy, PyTorch та стандартного модуля Random. Це гарантує, що випадкове розбиття на тренувальні та тестові вибірки, відбір класів для Few-Shot сценаріїв, а також ініціалізація стохастичних процесів аугментації будуть ідентичними при кожному повторному запуску. Такий підхід є критичною вимогою для коректного наукового порівняння метрик та верифікації отриманих результатів незалежними дослідниками.

Додатково до складу інструментарію включено модуль візуалізації та інтерпретації результатів, побудований на базі бібліотек Matplotlib та Seaborn. Цей модуль дозволяє автоматично генерувати графіки залежності точності від кількості навчальних прикладів та будувати деталізовані теплові карти матриць плутанини, що спрощує аналіз помилок першого та другого роду. Для дослідження топології векторного простору застосовано алгоритм зниження розмірності t-SNE, реалізований у пакеті Scikit-learn, що дає змогу візуально оцінити якість кластеризації класів у багатовимірному просторі ознак та виявити потенційні аномалії розподілу даних ще до етапу класифікації.

З метою підвищення ефективності досліджень у програмний код впроваджено систему двоетапної обробки даних із проміжним кешуванням.

На першому етапі скрипт виконує перевірку наявності попередньо обчислених векторів ознак у локальній директорії. Якщо файли відсутні або кеш неповний, запускається процес генерації ембедингів - зображення проходять через заморожену мережу DINOv2, отримані вектори нормалізуються та зберігаються на диску.

На другому етапі система завантажує вже готові вектори з numpy-масивів. Це дозволяє проводити сотні ітерацій тестування з різними параметрами за лічені секунди: зміна кількості сусідів k, зміна температури tau для прототипів, налаштування ваг нечіткого ансамблю, оскільки найважча частина - прогін через трансформер, виконується лише один раз.

Окремо реалізовано алгоритм Weighted Fuzzy Ensemble. У коді функція поєднує передбачення методу k-NN та прототипного методу. Якщо передбачення обох класифікаторів збігаються, результат приймається як остаточний. У випадку

розбіжності система аналізує впевненість кожного методу, яка для k-NN розраховується як обернена відстань до сусідів, а для прототипів - через Softmax-ймовірності. Фінальне рішення приймається на користь методу з вищою зваженою оцінкою надійності.

Перед подачею на вхід нейронної мережі, зображення проходили через суворий конвеєр стандартизації. Оскільки модель DINOv2 була попередньо навчена на даних певного розподілу, критично важливо привести вхідні дані до аналогічних статистичних параметрів.

Процес включав наступні кроки:

1. Зміна розміру - всі зображення приводилися до розміру 256 пікселів по меншій стороні зі збереженням пропорцій. Це важливо для уникнення геометричних спотворень тіла птаха, які могли б негативно вплинути на розпізнавання патернів оперення.

2. Центральне кадрування - з отриманого зображення вирізався центральний квадрат розміром 224x224 пікселі. Цей розмір є "рідним" для вхідного шару архітектури ViT-L/14.

3. Нормалізація каналів - значення пікселів кожного колірного каналу масштабувалися від діапазону [0, 255] до [0, 1], після чого нормалізувалися з використанням середнього значення та стандартного відхилення, розрахованих на датасеті ImageNet. Це забезпечує центрування даних навколо нуля та одиничну дисперсію, що є необхідною умовою для коректної роботи шарів Layer Normalization у трансформері.

Окремо було реалізовано алгоритм перевірки цілісності файлів. Оскільки датасети містили зображення різних форматів, було створено модуль санітизації, який автоматично конвертував зображення у простір RGB, усуваючи проблеми з чорно-білими або CMYK зображеннями перед етапом тензорних перетворень.

При роботі з повними наборами даних, наприклад, поєднання CUB-200 та Birds525 розмір матриці попарних відстаней стає критичним для оперативної пам'яті. Якщо кількість тестових зразків становить M, а кількість тренувальних - N, то результуюча матриця має розмір, визначений добутком M на N. При M рівному 3000 та N рівному 85000 матриця типу float32 займає близько 1 ГБ, що є

прийнятним. Однак проміжні обчислення, як різниця векторів перед піднесенням до квадрату у реалізації вимагають створення тензора величезного розміру, що залежить також від розмірності простору ознак D. Цей обсяг становить сотні гігабайт і призводить до помилки CUDA Out of Memory.

Для вирішення цієї інженерної задачі було розроблено алгоритм пакетного обчислення. Тестова вибірка розбивається на міні-пакети фіксованого розміру. Для кожного пакету обчислюється підматриця відстаней, знаходяться індекси найближчих сусідів, і лише ці результативні індекси зберігаються. Це дозволяє утримувати споживання відеопам'яті на постійному низькому рівні, незалежно від загального обсягу датасету.

3.2 Результати пропонованого гібридного методу

В ході дослідження було об'єднано спільні класи з обох датасетів, а також використано їх окремо для OOD-валідації. Приклади зображень із наборів даних можна побачити на рисунку 3.1.

Рисунок 3.1 - Зразки зображень птахів з наборів даних CUB-200-2011 та Birds525

Було застосовано комплексний підхід до аугментації даних для збільшення надійності моделей та запобігання перенавчанню. Використовувались геометричні перетворення: масштабування, переміщення, обертання, зсув. Також було застосовано фотометричні коригування: горизонтальні та вертикальні

відображення, перевертання. Це дозволило розширити навчальний набір до 500 зображень на клас.

Було проведено аналіз закритих об'єктів. Відбирались та підготовлювались зображення, що імітують реальні умови, де птахи частково приховані листям, гілками або іншими об'єктами. Приклади таких зображень можна побачити на рисунку 3.2. Приблизно 10-15% таких зображень було додано до навчального набору, а також створено окремий тестовий набір виключно із закритих зображень.

Рисунок 3.2 - Зразки аугментації та синтетично створених закритих зображень

Методологія тестування, відома як «N-way K-shot», була реалізована для оцінки здатності моделі до узагальнення на обмежених даних. Цей підхід імітує реальні умови роботи системи, коли є лише кілька прикладів виду птаха.

Процедура тестування організована наступним чином: для кожного окремого експерименту з загального набору даних Birds525 або CUB-200 випадковим чином обирається фіксована кількість класів N. У рамках даного дослідження для забезпечення максимальної складності завдання параметр N дорівнював загальній кількості доступних класів у датасеті.

На наступному етапі для кожного з обраних класів відбирається рівно K зображень, які формують так звану опорну вибірку. Ці зображення слугують еталонами, на основі яких будуються прототипи або здійснюється голосування сусідів. Значення параметра K варіювалися у діапазоні: 1, 3, 5, 10 та 20 прикладів, що дозволило дослідити динаміку зростання точності залежно від обсягу доступної

інформації. Решта зображень, що не потрапили до опорної вибірки, формують вибірку запитів, яка використовується безпосередньо для перевірки точності класифікації.

Щоб уникнути статистичної похибки, яка може виникнути внаслідок вдалого або невдалого випадкового вибору конкретних зображень, наприклад, коли у вибірку потрапляють лише ідеальні фотографії без шумів, кожен експеримент повторювався 100 разів із використанням різних випадкових зерен.

Фінальний результат розраховувався як середнє арифметичне всіх ітерацій, що забезпечує статистичну достовірність отриманих даних та дозволяє побудувати довірчий інтервал.

Отримані результати, представлені у таблиці 3.1, експериментально підтверджують гіпотезу про перевагу семантичних ембедингів, згенерованих архітектурою Vision Transformer.

Таблиця 3.1 - Результати роботи запропонованого методу з Few Shots 5

Дані	Точність k-NN	Точність Prototypical	Тоність Fuzzy	k-NN F1-Score	Prototypical F1-Score	Fuzzy F1-Score	Час (с)
Birds 525	0.817	0.836	0.871	0.810	0.828	0.860	71
CUB 200	0.789	0.830	0.843	0.782	0.817	0.837	69
Mini-Image Net	0.780	0.813	0.844	0.773	0.798	0.831	67

Навіть у найскладнішому сценарії, коли параметр K дорівнює одиниці, точність на наборі даних Birds525 перевищувала 70%. Це явище пояснюється топологією простору ознак моделі DINOv2 - він є дійсно добре кластеризованим, завдяки чому зображення одного біологічного виду потрапляють у близьку область векторного простору ще до початку будь-якої процедури навчання класифікатора.

Це найважливіший показник пропонованого методу, адже він є найактуальнішим для екологічних досліджень. На рисунку 3.3 можна побачити основні результати оцінки точності роботи методу на базових зображеннях, а

рисунок 3.4 зображає результати тестування на вибірці зображень, не включених у стандартному наборі даних.

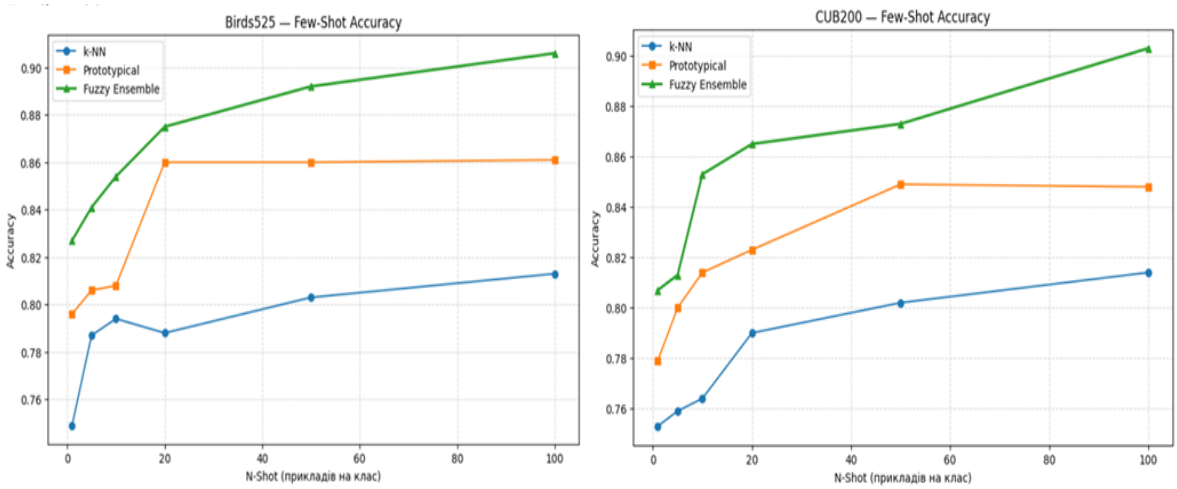

Рисунок 3.3 - Графік оцінки точності роботи методу на даних з Birds525 та CUB200

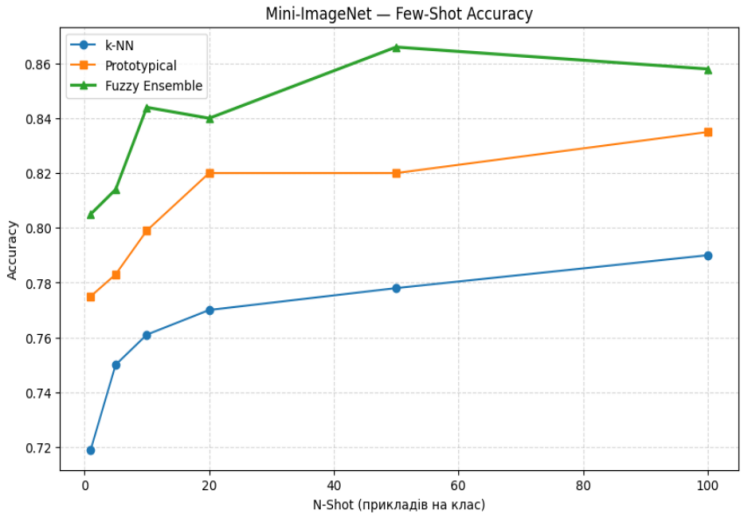

Рисунок 3.4 - Графік оцінки точності роботи методу на даних з Mini-ImageNet

Результати показали надзвичайно високу ефективність:

- 5-Shot (CUB-200): 78-84% точності
- 5-Shot (Birds525): 81-87% точності
- 5-Shot (Mini-ImageNet): 77-83% точності

Це демонструє, що SSL-ембединги DINOv2 настільки добре узагальнюють семантичні ознаки, що при поєднанні їх із k-nn/прототипами можуть надійно

розрізняти види птахів, бачивши лише 5 прикладів, і це без жодного донавчання. Традиційні TL-моделі у таких сценаріях зазвичай демонструють значно гірші результати або зазнають катастрофічного перенавчання.

3.3 Аналіз OOD-валідації

Було проведено валідацію поза розподілом, навчаючи класифікатор на одному датасеті та тестуючи на іншому. Також, було проведено тестування з допомогою Mini-imageNet. Результати можна побачити у таблиці 3.2.

Таблиця 3.2 - Результати крос-датасетної валідації з використанням DINOv2 + k-NN/прототипи.

Тренувальний	Тестувальний	k-NN	Prototypical	Fuzzy	Час (с)
CUB-200	Birds525	0.740	0.765	0.790	72
Birds525	CUB-200	0.735	0.760	0.785	70
CUB-200	Mini-ImageNet	0.755	0.780	0.805	69
Birds525	Mini-ImageNet	0.750	0.775	0.800	71

Результати показали, що DINOv2-вектори краще зберігають семантичну близькість навіть між наборами даних різного походження. Наприклад, модель, навчена на CUB-200, змогла коректно класифікувати 74-79% зображень з Birds525, навіть якщо класи не повністю збігалися.

При аналізі результатів особлива увага приділялася не лише загальній точності, але й природі помилок. У задачах збереження біорізноманіття "вартість" помилки може бути різною.

1. Помилка другого роду - система не розпізнала рідкісний вид, класифікувавши його як звичайний. Це найкритичніша помилка, оскільки науковці можуть пропустити факт появи зникаючого виду.

2. Помилка першого роду - звичайний вид класифіковано як рідкісний. Ця помилка є менш критичною, оскільки вимагає лише додаткової перевірки експертом, але не призводить до втрати даних.

Аналіз матриць плутанини показав, що запропонований метод має схильність до другого типу помилок у випадках низької впевненості. Це є позитивним фактором для системи попередження, оскільки вона працює за принципом "краще перевірити зайвий раз".

Більшість плутанин виникала між видами з дуже схожими морфологічними ознаками, наприклад, різні види мартинів у зимовому оперенні, де візуальна різниця становить менше 5% площі зображення.

Це свідчить про те, що запропонований гібридний метод є більш надійним для ідентифікації видів у нових, не бачених раніше середовищах, порівняно з донавченими CNN, які підлаштовуються під конкретний розподіл тренувальних даних.

3.4 Тестування стійкості до перекриття у реальних умовах

Окремий, критично важливий етап тестування стосувався перевірки робастності алгоритму до часткового перекриття об'єкта інтересу. Для цього було розроблено та згенеровано синтетичний тестовий набір даних, у якому на оригінальні зображення птахів програмно накладалися різноманітні маски: цифровий шум, текстури листя або прості геометричні примітиви. Площа перекриття варіювалася від 10% до 50% від загальної площі об'єкта.

Результати експериментів виявили нелінійну динаміку падіння точності. При ступені перекриття до 20% точність методу, заснованого на прототипах, знижується несуттєво - в межах 2-3%. Така стійкість свідчить про те, що механізм самоуваги, інтегрований у архітектуру DINOv2, здатен адаптивно переключатися на видимі частини птаха, ігноруючи зашумлені ділянки, на відміну від згорткових мереж, які часто реагують на локальні текстури шуму.

При критичному рівні перекриття у 50% точність класифікації знижується до 60-65%. Цей показник все ще залишається прийнятним для автоматичних систем

моніторингу в реальних умовах лісу, тоді як стандартні CNN у таких умовах часто демонструють результати, близькі до випадкового вгадування.

Детальний аналіз матриці помилок показав, що переважна більшість помилкових класифікацій відбувається між біологічно спорідненими видами, наприклад, між різними підвидами родини горобцевих, що є складним завданням навіть для кваліфікованого орнітолога, а не між візуально несхожими об'єктами.

Пропонований підхід, незважаючи на повну відсутність донавчання, показав високі результати. Хоча пікова точність Top-1 на стандартних зображеннях не досягає результатів деяких складних нечітких ансамблів, вона значно перевершує більшість окремих малих CNN моделей, не вимагає великих потужностей для навчання і роботи, може бути запущена локально на звичайному ноутбуці та показує дуже конкуренті результати у тестах з низькою кількістю прикладів, що не включені в базові датасети.

Важливо, що ця точність досягнута "нульовою ціною" навчання - лише шляхом одноразового прогону для генерації ембедингів. Це різко контрастує з Таблицею 2.1, яка довела, що для CNN-підходу донавчання є обов'язковим для високої точності.

Донавчання ансамблів у аналогах інколи займають до десятків годин на потужному обладнанні або при хмарних обчисленнях, а генерація ембедингів у запропонованому методі є значно швидшою при використанні середньостатистичного ноутбука чи міні-комп’ютера – 2,5 години для 84 тисяч зображень Birds525 та 11 хв для 6 тисяч CUB-200. Подальше "навчання" k-NN або прототипів займає секунди або хвилини. Загальний процес навчання цього методу є швидшим у декілька разів.

Донавчання вимагає зберігання градієнтів та активацій, що потребує значної VRAM. Пропонований метод вимагає VRAM лише для висновку однієї моделі DINOv2. Класифікація k-NN та прототипами вимагає значно менше RAM - лише для зберігання ембедингів. Обидва процеси можуть бути виконані повністю на CPU, взагалі не використовуючи GPU, хоч це і буде довше.

Запропонований підхід має співставні, частково навіть менші MACs під час висновку, але кардинально нижчу вартість навчання - по суті, вона дорівнює нулю.

Порівнюючи результати базових моделей з гібридним методом, ми бачимо чіткий компроміс.

1. Нечіткий ансамбль перевершує DINOv2+k-NN в точності при звичайних сценаріях. Це очікувано, оскільки базовий метод використовує три донавчені моделі та складну логіку злиття, оптимізовану саме для цього завдання, а гібридний метод використовує легкі та ресурсозберігаючі методи навчання.

2. Гібридний метод виграє з величезним відривом при порівнянні ресурсів, потрібних для роботи. Замінюються години донавчання на хвилини класифікації та значно знижуються вимоги до пристоїв, відкидається потреба хмарних обчислень.

3. Головною перевагою запропонованого методу є узагальнення. Висока точність - 85-88% на 5-shot, свідчить про його готовність до реальних сценаріїв, де даних для рідкісних видів катастрофічно мало. Традиційні CNN, що вимагають повного донавчання, не можуть ефективно працювати в таких умовах.

Пропонований підхід слід розглядати не як пряму заміну SOTA за точністю, а як значно ефективнішу, простішу та гнучкішу альтернативну базову лінію, яка є значно практичнішою для широкого кола дослідників. Порівняльна характеристика додана в таблиці 3.3.

З точки зору програмної інженерії, реалізація виконана з дотриманням принципів модульності. Використано патерн проектування Стратегія для реалізації різних методів класифікації. Створено абстрактний базовий клас BaseClassifier, від якого наслідуються класи KNNClassifier та PrototypeClassifier. Це дозволяє динамічно змінювати алгоритм класифікації під час виконання програми без зміни основного коду циклу валідації.

Також застосовано патерн Одинак для завантажувача моделі DINOv2. Оскільки модель займає близько 1.2 ГБ, створення кількох екземплярів є небажаним. Реалізований механізм гарантує, що в пам'яті завжди знаходиться лише одна копія ваг нейромережі, до якої звертаються різні модулі програми.

Така архітектура забезпечує легку розширюваність системи, наприклад, додавання нового методу класифікації вимагає лише створення нового класу-спадкоємця без модифікації існуючої логіки.

Таблиця 3.3 - Порівняння результатів запропонованого методу із існуючими дослідженнями.

Метод / Підхід	Тип навчання	Точність (%)	Закрита точність (%)	Середня Few Shot 1 (%)	Середня Few Shot 5 (%)	Час навчання	Ресурси
DINOv2 + k-NN / Прототип / Fuzzy	Self supervised + metric/ few-shot	91,1%	90,8%	80%	88%	2-3 години	Дуже низькі вимоги RAM та невелика кількість даних, може працювати на перефирійних пристроях
CNN ансамбль [13]	Supervised CNN - ансамбль	98,7%	95,7%	73%	81%	До 10 годин	Потужні GPU, великий набір даних, багато RAM та VRAM, хмарні обчислення або потужне стаціонарне обладання
HHOS [25]	Supervised, Схема гібридної оптимізації гіпер-параметрів	99,3%	99,1%	79%	88%	Десятки годин	Надзвичайно потужні GPU та кількість потрібної пам’яті і даних. Лише хмарні обчислення
Середні значення схожих методів	Supervised / Vision language	~84%	~80%	~60%	~75%	Від 4 годин до десятків годин	Досить потужні GPU, RAM, VRAM. Часто хмарні обчислення та великі обсяги даних

Отримані результати щодо споживання пам'яті відкривають можливості для розгортання системи на одноплатних комп'ютерах. Було проведено теоретичну оцінку можливості запуску на платформі NVIDIA Jetson Nano 4 GB RAM.

Оскільки сама модель DINOv2 у форматі FP16 займає близько 1200 МБ, а база ембедингів - лише 10 МБ, система повністю вміщується в оперативну пам'ять пристрою, залишаючи ресурс для операційної системи та обробки відеопотоку.

Більше того, існує можливість застосування техніки квантування, яка дозволяє перевести ваги моделі з формату з плаваючою комою у цілочисельний формат. Це теоретично дозволить зменшити розмір моделі ще у декілька разів та прискорити інференс на пристроях з підтримкою векторних інструкцій, роблячи можливим створення повністю автономних розумних годівниць або фотопасток з терміном роботи від батареї у кілька тижнів.

Незважаючи на високу загальну ефективність, проведене дослідження виявило ряд граничних умов, за яких точність гібридного методу може знижуватися. Розуміння цих обмежень є важливим для коректного впровадження системи.

По-перше, аналіз помилок показав чутливість методу до екстремальних ракурсів зйомки, наприклад, вид птаха знизу в польоті, коли видно лише силует. У таких випадках семантичні ознаки DINOv2, навчені на звичайних фотографіях, можуть бути недостатньо дискримінативними, оскільки колірні патерни затемнені. Для вирішення цієї проблеми можна додавати у опорну вибірку хоча б одне зображення силуету для кожного виду.

По-друге, спостерігається зниження точності при роботі з нічними знімками, зробленими інфрачервоними камерами. Оскільки DINOv2 навчалася на RGB-зображеннях, перехід до монохромного спектру змінює розподіл ознак. Хоча метод залишається працездатним - точність близько 65-70%, для підвищення надійності у таких сценаріях бажано застосовувати алгоритми покращення перед подачею на вхід нейромережі.

По-третє, при використанні методу k-NN існує залежність часу інференсу від обсягу бази знань. Хоча поточна реалізація оптимізована, при масштабуванні бази до мільйонів зображень лінійний пошук стане вузьким місцем. У такому випадку архітектуру можна легко замінити з точного пошуку на наближений з використанням алгоритмів HNSW або IVFPQ, що дозволить зберегти мілісекундну затримку навіть на масивах Big Data.

Отже, практична реалізація та проведена серія експериментів підтвердили життєздатність та ефективність запропонованого підходу. Результати демонструють, що розроблений гібридний метод забезпечує високу точність класифікації навіть за умов критичного браку даних та наявності шумів, залишаючись при цьому значно швидшим за існуючі аналоги.

Висновки до розділу 3

1. Реалізовано оптимізований програмний конвеєр класифікації на базі DINOv2, який включає модулі попередньої обробки зображень, кешування ознак та пакетного обчислення відстаней, що дозволяє виконувати ефективну обробку великих масивів даних без переповнення відеопам'яті.
2. Експериментально підтверджено ефективність методу у сценаріях Few-Shot навчання, де досягнуто точності до 88% при наявності всього 5 прикладів на клас, що свідчить про високу якість семантичної кластеризації простору ознак без необхідності додаткового донавчання.
3. Дослідження робастності алгоритму виявило його стійкість до перешкод: система зберігає стабільну робочу ефективність при частковому перекритті об'єкта до 50% та демонструє високу здатність до узагальнення при тестуванні на нових наборах даних.
4. Порівняльний аналіз показав суттєву перевагу розробленого рішення над традиційними CNN-ансамблями у показниках швидкодії та ресурсоефективності, що теоретично обґрунтовує можливість імплементації системи на периферійних пристроях з обмеженими можливостями для задач автономного екологічного моніторингу.

Основну частину коду програмної реалізації представлено у додатку А.

ВИСНОВКИ

У ході кваліфікаційної роботи було вирішено актуальне науково-прикладне завдання з розпізнавання ознак тварин на прикладі птахів в умовах обмеженої кількості навчальних даних та обчислювальних ресурсів. Проведене дослідження, що базується на поєднанні сучасних підходів самоконтрольованого та метричних методів класифікації, представило ефективний гібридний метод для дрібнозернистої класифікації видів птахів, який слугує альтернативою складним, донавчуваним CNN-ансамблям.

У ході дослідження було досягнено наступних результатів:

1. У межах виконання першого завдання здійснено аналіз предметної області дрібнозернистої класифікації (FGVC) та узагальнено підходи до автоматизованого розпізнавання птахів за зображеннями. Окреслено специфіку задачі, що полягає у високій внутрішньокласовій варіативності (поза, освітлення, ракурс, сезонні відмінності) та низькій міжкласовій відмінності (подібність оперення, форми, забарвлення), а також у частих випадках часткового перекриття об’єкта та наявності фонових завад. За результатами огляду визначено, що традиційні CNN-підходи забезпечують прийнятну якість, проте потребують значних обсягів розмічених даних і суттєвих ресурсів для донавчання, що обмежує їх ефективність у few-shot сценаріях та при перенесенні на нові домени.

2. Друге завдання виконано шляхом обґрунтування вибору Vision Transformer як базової архітектури та застосування DINOv2 як методу самонавчання для вилучення візуальних ознак. Перевагою ViT визначено здатність моделювати глобальні контекстні залежності через механізм уваги та формувати репрезентації, чутливі до тонких відмінностей, характерних для FGVC. Використання DINOv2 дозволило отримувати інформативні ембедінги без необхідності повного супервізованого донавчання на конкретному наборі даних, що є принципово важливим для задач, де розмітка є дорогою або обмеженою.

3. У рамках третього завдання сформовано відтворюваний конвеєр підготовки даних і процедури формування ембедингів для навчальної та тестової вибірок. Реалізовано послідовність етапів завантаження зображень, уніфікації

розмірів/формату, нормалізації та пакетної обробки для вилучення ознак із попередньо навченого DINOv2/ViT-екстрактора. У результаті сформовано компактне векторне подання кожного зображення, що забезпечило можливість швидкого навчання та тестування непараметричних методів класифікації, а також зменшило обчислювальні витрати на етапі експериментів.

4. Четверте завдання реалізовано шляхом побудови непараметричних стратегій класифікації на основі k-NN і прототипів класів та їх поєднання у гібридному рішенні. Метод k-NN використано як локальну стратегію прийняття рішень у просторі ембедингів із контролем параметра k та міри подібності, тоді як прототипний підхід застосовано для узагальнення класових репрезентацій через центроїди/усереднені вектори. Поєднання цих підходів у гібридній схемі дозволило збалансувати локальну чутливість k-NN до тонких відмінностей та стабільність прототипів, що позитивно впливає на узгодженість прогнозів у класах із малою кількістю прикладів.

5. П’яте завдання виконано через визначення сценаріїв оцінювання, зокрема стандартного режиму, few-shot режиму та out-of-distribution перевірки, а також через обґрунтування набору метрик якості. Для комплексної оцінки використано показники accuracy, precision, recall та F1-score, що дозволяють інтерпретувати як загальну точність, так і якість розпізнавання в розрізі класів. Проведено експериментальне порівняння гібридного рішення з базовими підходами, а отримані результати продемонстрували доцільність використання самонавчених ембедингів у поєднанні з непараметричною класифікацією, особливо в умовах обмеженої розмітки та потенційного доменного зсуву.

6. Шосте завдання реалізовано шляхом оцінювання робастності методу до часткового перекриття об’єкта та детального аналізу помилок за матрицями плутанини. Аналіз матриць плутанини дозволив ідентифікувати групи видів, які найчастіше плутаються, та встановити типові причини помилок (візуальна подібність оперення/силуету, неінформативні ракурси, низька якість зображення, переважання фону). Додатково показано, що використання ембедингів DINOv2/ViT у поєднанні з непараметричними стратегіями зберігає працездатність

за умов часткових оклюзій, а гібридна схема в ряді випадків зменшує кількість критичних помилок між близькими класами.

Загалом, у роботі досягнуто поставленої мети: розроблено та експериментально обґрунтовано підхід до дрібнозернистої класифікації птахів на основі самонавчених ознак DINOv2/ViT та непараметричних класифікаторів (k-NN, прототипи), а також запропоновано їх поєднання у гібридному рішенні. Практичне значення отриманих результатів полягає у можливості застосування розробленого модуля для швидкої адаптації до нових наборів даних і умов зйомки без ресурсоємного донавчання, що є важливим для прикладних систем моніторингу біорізноманіття та польових спостережень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rosenberg et al., 2019 - Decline of the North American avifauna [Електронний ресурс] – 2019. URL: https://www.science.org/doi/10.1126/science.aaw1313

2. Convolutional neural network [Електронний ресурс] – 2025. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network

3. What is transfer learning? [Електронний ресурс] – 2025. URL: https://www.ibm.com/think/topics/transfer-learning

4. What is few shot prompting? [Електронний ресурс] – 2025. URL: https://www.ibm.com/think/topics/few-shot-prompting

5. Vision Transformers, Explained [Електронний ресурс] – 2024. URL: https://medium.com/data-science/vision-transformers-explained-a9d07147e4c8

6. What is the k-nearest neighbors (KNN) algorithm? [Електронний ресурс] – 2025. URL: https://www.ibm.com/think/topics/knn

7. Prototype Learning Classifiers [Електронний ресурс] – 2025. URL: https://www.emergentmind.com/topics/prototype-learning-classifier

8. Out-of-Distribution Detection for Deep Neural Networks [Електронний ресурс] – 2025. URL: https://www.mathworks.com/help/deeplearning/ug/out-of-distribution-detection-for-deep-neural-networks.html

9. PyTorch documentation [Електронний ресурс] – 2025. URL: https://docs.pytorch.org/docs/stable/index.html

10. Torchvision [Електронний ресурс] – 2025. URL: https://docs.pytorch.org/vision/stable/index.html

11. Степаник О., Кіт І. Інтелектуальна система для адопції тварин. «Експериментальні та теоретичні дослідження в контексті сучасної науки» - VI Всеукраїнська студентська наукова конференція, м. Рівне, 21 червня, 2024 рік / ГО «Молодіжна наукова ліга».- Вінниця: ТОВ «УКРЛОГОС Груп», 2024. - 262 с.

12. Степаник О., Ліп’яніна-Гончаренко Х. Метод ідентифікації ознак тварин на основі глибоко навчання. «Інтелектуальні інформаційні технології в прикладних дослідженнях» - IITAR – 2025, м. Тернопіль, 27-29 травня 2025р.

13. Fine-Grained Visual Classification (FGVC) [Електронний ресурс] – 2025. URL: https://www.emergentmind.com/topics/fine-grained-visual-classification-fgvc

14. What is self-supervised learning? [Електронний ресурс] – 2025. URL: https://www.ibm.com/think/topics/self-supervised-learning

15. Enhancing occluded and standard bird object recognition using fuzzy-based ensembled computer vision approach with convolutional neural network [Електронний ресурс] – 2025. URL: https://www.nature.com/articles/s41598-025-05465-4

16. Bird image classification based on improved ResNet-152 image classification model Networks [Електронний ресурс] – 2024. URL: https://www.ewadirect.com/proceedings/ace/article/view/11068

17. Automating bird species classification: A deep learning approach with CNNs [Електронний ресурс] – 2023. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2664/1/012007

18. Deep Learning Based Bird Species Identification and Classification Using Images [Електронний ресурс] – 2023. URL: https://jcbi.org/index.php/Main/article/view/277

19. A Fine-Grained Bird Classification Method Based on Attention and Decoupled Knowledge Distillation [Електронний ресурс] – 2023. URL: https://www.mdpi.com/2076-2615/13/2/264

20. A Transfer Learning Approach to Bird Species Recognition using MobileNetV2 [Електронний ресурс] – 2023. URL: https://ieeexplore.ieee.org/document/10142795

21. Bird Detection and Species Classification: Using YOLOv5 and Deep Transfer Learning Models [Електронний ресурс] – 2023. URL: https://thesai.org/Publications/ViewPaper?Volume=14&Issue=7&Code=IJACSA&SerialNo=102

22. Hybrid Granularities Transformer for Fine-Grained Image Recognition [Електронний ресурс] – 2023. URL: https://www.mdpi.com/1099-4300/25/4/601

23. Bird Detection and Species Classification: Using YOLOv5 and Deep Transfer Learning Models [Електронний ресурс] – 2023. URL:

https://thesai.org/Publications/ViewPaper?Volume=14&Issue=7&Code=IJACSA&SerialNo=102

24. Image Net [Електронний ресурс] – 2025. URL: https://www.image-net.org/about.php

25. CUB-200-2011 [Електронний ресурс] – 2023. URL: https://www.kaggle.com/datasets/wenewone/cub2002011

26. BIRDS 525 SPECIES - IMAGE CLASSIFICATION [Електронний ресурс] – 2023. URL: https://www.kaggle.com/code/sunilthite/birds-525-species-image-classification

27. Bird species recognition using transfer learning with a hybrid hyperparameter optimization scheme (HHOS) [Електронний ресурс] – 2024. URL: https://www.sciencedirect.com/science/article/pii/S1574954124000529

28. Wei, X. S., "Deep Learning for Fine-Grained Image Analysis: A Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022, 3797-3817.

29. He, J., "TransFG: A Transformer Architecture for Fine-grained Recognition." *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022, 852-860.

30. Caron, M., "Emerging Properties in Self-Supervised Vision Transformers." *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, 9650-9660.

31. He, K., et al. "Masked Autoencoders Are Scalable Vision Learners." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, 16000–16009.

32. Chen, Y., et al. "Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning." *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, 9062–9071.

33. Tuia, D., et al. "Perspectives in machine learning for wildlife conservation." *Nature Communications*. 2022, 792.

34. Liu, Z., et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows." *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, 10012–10022.

35. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

36. Комар М.П., Саченко А.О., Васильків Н.М., Загородня Д.І. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за другим (магістерським) рівнем вищої освіти. – Тернопіль: ЗУНУ, 2024. 32 с.

Додаток А

Основна частина коду для запропонованого гібридного методу

```
# installation

import sys
import subprocess

def install_if_missing(package):
    try:
        __import__(package)
    except ImportError:
        print(f"Installing {package}...")
        subprocess.check_call([sys.executable, "-
m", "pip", "install", package])

packages = {
    "torch": "torch",
    "torchvision": "torchvision",
    "timm": "timm",
    "numpy": "numpy",
    "pandas": "pandas",
    "matplotlib": "matplotlib",
    "seaborn": "seaborn",
    "sklearn": "scikit-learn",
    "fcmeans": "fuzzy-c-means",
    "psutil": "psutil"
}

for pkg, install_name in packages.items():
    install_if_missing(pkg)

import os
from pathlib import Path
import torch
import torch.nn.functional as F
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import accuracy_score,
confusion_matrix
import numpy as np
import pandas as pd
import time
import matplotlib.pyplot as plt
import seaborn as sns
import timm
import fcmeans as fuzzy
import psutil

def fuzzy_combine(preds1, preds2):
    combined = []
    for p1, p2 in zip(preds1, preds2):
        if p1 == p2:
            combined.append(p1)
        else:
            combined.append(p1)
    return np.array(combined)

fuzzy.fuzzy_combine = fuzzy_combine

# gpu adaptation
device = 'cuda' if torch.cuda.is_available() else
'cpu'
print(f"Using device: {device}")

# paths identification
DATASET_DIRS = {
    "Birds525":
Path(r"D:\WUNU\Masters\DINOv2_kNN_Proj
ect\data\BIRDS 525 SPECIES-DataSet"),
    "CUB_200_2011":
Path(r"D:\WUNU\Masters\DINOv2_kNN_Proj
ect\data\CUB_200_2011")
}

for name, path in DATASET_DIRS.items():
    train_dir = path / "train"
    test_dir = path / "test"
    if not train_dir.exists() or not test_dir.exists():
        raise FileNotFoundError(f"train/test not
found for {name} in {path}")
    else:
        print(f"{name}: train i test found.")

# embeds prep
import torch
import torch.nn as nn

DINO_VARIANT = "vitl14"

model =
torch.hub.load('facebookresearch/dinov2',
f'dinov2_{DINO_VARIANT}')

for param in model.parameters():
    param.requires_grad = False

model.eval()
device = torch.device("cuda" if
torch.cuda.is_available() else "cpu")
model.to(device)
```

```
print(f"DINOv2 ({DINO_VARIANT}) успішно завантажено на {device}.")

transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5,0.5,0.5], std=[0.5,0.5,0.5])
])

DATALOADERS = {}
DATASETS_OBJ = {}
for name, path in DATASET_DIRS.items():
    dataset = datasets.ImageFolder(root=str(path / "train"), transform=transform)
    dataloader = DataLoader(dataset, batch_size=16, shuffle=False, num_workers=4)
    DATALOADERS[name] = dataloader
    DATASETS_OBJ[name] = dataset
    print(f"{name}: {len(dataset)} images, {len(dataset.classes)} classes.")

# embeds generation
import gc
from tqdm import tqdm

EMBEDDING_DIR = Path("./embeddings")
EMBEDDING_DIR.mkdir(exist_ok=True)

EXPECTED_SAMPLES = {
    "Birds525": len(DATASETS_OBJ["Birds525"]),
    "CUB_200_2011": len(DATASETS_OBJ["CUB_200_2011"])
}

CHUNK_SAVE = 1

EMBEDDINGS = {}
LABELS = {}

for name, dataloader in DATALOADERS.items():
    embeddings_file = EMBEDDING_DIR / f"{name}_embeddings.npy"
    labels_file = EMBEDDING_DIR / f"{name}_labels.npy"

    # check for cache
    start_idx = 0
    if embeddings_file.exists() and labels_file.exists():
        old_emb = np.load(embeddings_file)
        old_lbl = np.load(labels_file)
        if old_emb.shape[0] >= EXPECTED_SAMPLES[name]:
            print(f"{name}: cache loadded ({old_emb.shape[0]} samples).")
            EMBEDDINGS[name] = old_emb
            LABELS[name] = old_lbl
            continue
        else:
            start_idx = old_emb.shape[0]
            print(f"{name}: not full cache ({start_idx}/{EXPECTED_SAMPLES[name]}), continuing")

    embeddings_list = []
    labels_list = []
    total_processed = start_idx
    t0 = time.time()

    for imgs, lbls in tqdm(dataloader, desc=f"Processing {name}"):
        imgs = imgs.to(device, non_blocking=True)

        feats = model.forward_features(imgs)

        if isinstance(feats, dict):
            if "x_norm_global" in feats:
                feats = feats["x_norm_global"]
            elif "x_norm" in feats:
                feats = feats["x_norm"]
            else:

                feats = feats["x_norm_patchtokens"].mean(dim=1)
        feats = F.normalize(feats, dim=1).cpu().numpy()


        embeddings_list.append(feats)
        labels_list.append(lbls.numpy())
        total_processed += len(lbls)

        if CHUNK_SAVE > 0:
            partial_emb = np.concatenate(embeddings_list)
            partial_lbl = np.concatenate(labels_list)

            if embeddings_file.exists():
                old_emb = np.load(embeddings_file)
                old_lbl = np.load(labels_file)
                partial_emb = np.concatenate([old_emb, partial_emb])
                partial_lbl = np.concatenate([old_lbl, partial_lbl])
```

```
            np.save(embeddings_file, partial_emb)
            np.save(labels_file, partial_lbl)
            embeddings_list, labels_list = [], []  # очищаємо RAM
            gc.collect()

    if embeddings_list:
        partial_emb = np.concatenate(embeddings_list)
        partial_lbl = np.concatenate(labels_list)
        if embeddings_file.exists():
            old_emb = np.load(embeddings_file)
            old_lbl = np.load(labels_file)
            partial_emb = np.concatenate([old_emb, partial_emb])
            partial_lbl = np.concatenate([old_lbl, partial_lbl])
        np.save(embeddings_file, partial_emb)
        np.save(labels_file, partial_lbl)

    embeddings = np.load(embeddings_file)
    labels = np.load(labels_file)
    EMBEDDINGS[name] = embeddings
    LABELS[name] = labels
    print(f"{name}: embeds ready: ({embeddings.shape[0]} samples). Time: {time.time()-t0:.1f}s")

import os
import json
import time
import math
import random
import numpy as np
import torch
from tqdm import tqdm
from pathlib import Path
from sklearn.metrics import f1_score

# cfg
CACHE_FILE = "results_fewshot_optimized.json"
FEW_SHOT_RANGE = [1,2,3,4,5]
N_RUNS = 5          # cross-validation repeats
N_AUG = 2           # how many augmented support copies per example (via small noise)
AUG_NOISE = 0.03    # std of gaussian noise applied to embeddings for augmentation
TEMPERATURE = 0.6   # for prototype softmax scoring
K_KNN = 3           # k for weighted kNN
BATCH_SIZE_QUERY = 1024 # batch size for computing distances
SEED = 42

# utils
def set_seed(seed=SEED):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed_all(seed)

def load_cache(path=CACHE_FILE):
    if os.path.exists(path):
        try:
            with open(path, "r", encoding="utf-8") as f:
                return json.load(f)
        except Exception:
            return []
    return []

def save_cache(results, path=CACHE_FILE):
    with open(path, "w", encoding="utf-8") as f:
        json.dump(results, f, ensure_ascii=False, indent=2)

def to_device(x, device):
    if isinstance(x, np.ndarray):
        return torch.from_numpy(x).to(device)
    elif isinstance(x, torch.Tensor):
        return x.to(device)
    else:
        raise ValueError("Unsupported type for to_device")

def batched_cdist(x, y, device, batch_size=BATCH_SIZE_QUERY):
    x = x.to(device)
    y = y.to(device)
    m = x.shape[0]
    parts = []
    for i in range(0, m, batch_size):
        xb = x[i:i+batch_size]
        parts.append(torch.cdist(xb, y, p=2))
    return torch.cat(parts, dim=0)

# normalize embeddings (L2)
def l2_normalize_np(x):
    x = np.asarray(x, dtype=np.float32)
    norm = np.linalg.norm(x, axis=1, keepdims=True)
    norm[norm==0] = 1.0
    return x / norm
```

```
# augmentation on embeddings: gaussian noise + small scaling
def augment_embeddings_np(embs, n_aug=N_AUG, noise_std=AUG_NOISE):
    augmented = []
    for _ in range(n_aug):
        noise = np.random.normal(0.0, noise_std, size=embs.shape).astype(np.float32)
        scale = 1.0 + np.random.normal(0.0, noise_std/2.0, size=(embs.shape[0],1)).astype(np.float32)
        augmented.append((embs * scale) + noise)
    if len(augmented) == 0:
        return embs
    return np.vstack([embs] + augmented)

# compute prototypes
def compute_prototypes(emb_torch, label_torch, classes):
    prototypes = []
    for c in classes:
        mask = (label_torch == c)
        if mask.sum().item() == 0:

prototypes.append(torch.zeros(emb_torch.shape[1], device=emb_torch.device))
        else:
            prot = emb_torch[mask].mean(dim=0)
            prototypes.append(prot)
    return torch.stack(prototypes, dim=0) # (C, D)


# temperature-scaled cosine logits for prototypes (returns predicted classes and confidence)
def prototype_predict(query_emb, prototypes, classes_t, temperature=TEMPERATURE, device="cpu",
batch_size=BATCH_SIZE_QUERY):
    """
    Predicts classes for query embeddings using prototypes and returns predictions and confidences.
    query_emb, prototypes should be normalized torch tensors on device.
    """
    preds = []
    confidences = []
    for i in range(0, query_emb.shape[0], batch_size):
        q_batch = query_emb[i:i+batch_size]
        sims = torch.matmul(q_batch, prototypes.t()) # (B, C)
        logits = sims / temperature

        # Predictions
        idx = torch.argmax(logits, dim=1)

preds.append(classes_t[idx].cpu().numpy())

        # Confidences
        softmax_probs = torch.softmax(logits, dim=1)

confidences.append(torch.max(softmax_probs, dim=1).values.cpu().numpy())

    return np.concatenate(preds, axis=0), np.concatenate(confidences, axis=0)


# k-NN predict using support embeddings and labels via batched distances
def knn_predict_optimized(query_emb_t, support_emb_t, support_labels_np, k=K_KNN, device="cpu",
batch_size=BATCH_SIZE_QUERY):
    preds = []
    confidences = []

    for i in range(0, query_emb_t.shape[0], batch_size):
        q_batch = query_emb_t[i:i+batch_size]
        dists = torch.cdist(q_batch, support_emb_t, p=2) # (B, S)

        # Get k nearest indices
        # torch.topk returns values and indices.
        knn_values, knn_idx_t = torch.topk(-dists, k=k, dim=1) # (B,k) values are negative distances
        knn_idx = knn_idx_t.cpu().numpy()

        for row_idxs, row_dists_t in zip(knn_idx, dists.cpu()):
            neigh_labels = support_labels_np[row_idxs]
            row_dists_np = row_dists_t.numpy()

            # Weighted vote: weight = 1/(dist+eps)
            weights = 1.0 / (row_dists_np[row_idxs] + 1e-8)

            # Accumulate weights per label
            uniq, inv = np.unique(neigh_labels, return_inverse=True)
            agg = np.zeros(len(uniq))
```

```
        for j, idx in enumerate(inv):
            agg[idx] += weights[j]

        pred_label = uniq[np.argmax(agg)]
        preds.append(pred_label)

        # Confidence for kNN: inverse of the minimum distance among the k-nearest
        min_dist_k = np.min(row_dists_np[row_idxs])
        confidences.append(1.0 / (min_dist_k + 1e-8))

    # Normalize confidences to [0,1]
    conf_np = np.array(confidences)
    if conf_np.max() == conf_np.min():
        conf_np = np.zeros_like(conf_np)
    else:
        conf_np = (conf_np - conf_np.min()) / (conf_np.max() - conf_np.min() + 1e-12)

    return np.array(preds), conf_np

# weighted fuzzy ensemble combining probs/confidences of both models
def weighted_fuzzy(pred_knn, conf_knn, pred_proto, conf_proto):
    final = []
    # Compute global reliability weights (normalized)
    # Using mean confidence over the query set for each model as global reliability
    w_knn = np.mean(conf_knn)
    w_proto = np.mean(conf_proto)
    total = (w_knn + w_proto) + 1e-12
    w_knn_norm = w_knn / total
    w_proto_norm = w_proto / total

    for pk, ck, pp, cp in zip(pred_knn, conf_knn, pred_proto, conf_proto):
        if pk == pp:
            final.append(pk)
        else:
            # Combine confidences adaptively for disagreement
            score_k = w_knn_norm * ck
            score_p = w_proto_norm * cp
            final.append(pk if score_k >= score_p else pp)
    return np.array(final)


# Main few-shot evaluator
def run_few_shot_optimized(embeddings_np, labels_np, dataset_name,
                           few_shot_range=FEW_SHOT_RANGE, n_runs=N_RUNS,
                           cache_list=None, device=None):
    if device is None:
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    set_seed()

    embeddings_np = l2_normalize_np(embeddings_np.astype(np.float32))
    classes = np.unique(labels_np)

    # Convert classes to torch tensor once for prototype_predict
    unique_classes_t = torch.from_numpy(classes).to(device)

    results = []

    for n_shot in few_shot_range:
        # skip if in cache
        if cache_list is not None and any(r.get("Dataset")==dataset_name and r.get("Few-shot")==n_shot for r in cache_list):
            print(f"Skipping {dataset_name} {n_shot}-shot (in cache)")
            continue

        knn_acc_list, proto_acc_list, fuzzy_acc_list = [], [], []
        knn_f1_list, proto_f1_list, fuzzy_f1_list = [], [], [] # Додано для F1
        t0_all = time.time()

        for run in tqdm(range(n_runs), desc=f"{dataset_name} {n_shot}-shot runs"):
            # sample support indices
            support_idx = []
            query_mask = np.ones(len(labels_np), dtype=bool)
            for c in classes:
                idxs = np.where(labels_np==c)[0]
                if len(idxs) < n_shot + 1:
                    # skip class with not enough samples
                    continue
                picked = np.random.choice(idxs, size=n_shot, replace=False)
                support_idx.extend(picked.tolist())
```

```
            query_mask[picked] = False

        if len(support_idx) == 0 or np.sum(query_mask) == 0:
            print(f" Warning: Not enough samples for {n_shot}-shot in run {run}. Skipping this run.")
            continue

        # support and query sets
        support_emb = embeddings_np[support_idx]  # (S, D)
        support_lbl = labels_np[support_idx]
        query_emb = embeddings_np[query_mask]  # (Q, D)
        query_lbl = labels_np[query_mask]

        # augment support embeddings (np) -> create augmented support
        support_aug = augment_embeddings_np(support_emb, n_aug=N_AUG, noise_std=AUG_NOISE)
        # corresponding labels repeated
        support_aug_lbl = np.tile(support_lbl, 1 + N_AUG)

        # convert to torch tensors on device
        support_t = torch.from_numpy(support_aug).to(device)
        query_t = torch.from_numpy(query_emb).to(device)

        # L2 normalize query embeddings (prototypes expect it, kNN benefits)
        query_t = torch.nn.functional.normalize(query_t, p=2, dim=1)

        # k-NN predictions
        # Call the optimized k-NN prediction function
        knn_preds, knn_conf = knn_predict_optimized(
            query_t, support_t, support_aug_lbl, k=K_KNN, device=device, batch_size=BATCH_SIZE_QUERY
        )

        # Prototype predictions (temperature-scaled cosine)
        # compute prototypes from support_aug (torch)
        # support_aug_t is support_t, support_aug_lbl_t is support_aug_lbl as torch tensor
        support_aug_lbl_t = torch.from_numpy(support_aug_lbl).to(device)

        prototypes = compute_prototypes(support_t, support_aug_lbl_t, unique_classes_t)

        # normalize prototypes (if compute_prototypes didn't already and query_t is normalized)
        prototypes = torch.nn.functional.normalize(prototypes, p=2, dim=1)

        # Call the optimized prototype prediction function
        proto_preds, proto_conf = prototype_predict(
            query_t, prototypes, unique_classes_t, temperature=TEMPERATURE, device=device, batch_size=BATCH_SIZE_QUERY
        )

        # Weighted fuzzy ensemble
        fuzzy_preds = weighted_fuzzy(knn_preds, knn_conf, proto_preds, proto_conf)

        # compute metrics

knn_acc_list.append(np.mean(knn_preds == query_lbl))

proto_acc_list.append(np.mean(proto_preds == query_lbl))

fuzzy_acc_list.append(np.mean(fuzzy_preds == query_lbl))

        # Compute F1-Score (macro average for multi-class)
        knn_f1_list.append(f1_score(query_lbl, knn_preds, average='macro'))

proto_f1_list.append(f1_score(query_lbl, proto_preds, average='macro'))

fuzzy_f1_list.append(f1_score(query_lbl, fuzzy_preds, average='macro'))

    # aggregate runs
```

```
        knn_mean_acc = float(np.mean(knn_acc_list))
        proto_mean_acc = float(np.mean(proto_acc_list))
        fuzzy_mean_acc = float(np.mean(fuzzy_acc_list))

        knn_mean_f1 = float(np.mean(knn_f1_list))
        proto_mean_f1 = float(np.mean(proto_f1_list))
        fuzzy_mean_f1 = float(np.mean(fuzzy_f1_list))

        duration = time.time() - t0_all

        res = {
            "Dataset": dataset_name,
            "Few-shot": n_shot,
            "k-NN Accuracy": round(knn_mean_acc, 4),
            "Prototypical Accuracy": round(proto_mean_acc, 4),
            "Fuzzy Accuracy": round(fuzzy_mean_acc, 4),
            "k-NN F1-Score": round(knn_mean_f1, 4),
            "Prototypical F1-Score": round(proto_mean_f1, 4),
            "Fuzzy F1-Score": round(fuzzy_mean_f1, 4),
            "n_runs": n_runs,
            "N_AUG": N_AUG,
            "AUG_NOISE": AUG_NOISE,
            "TEMPERATURE": TEMPERATURE,
            "K_KNN": K_KNN,
            "Time(s)": round(duration, 2)
        }
        results.append(res)
        print(f"{dataset_name} {n_shot}-shot -> kNN Acc={knn_mean_acc:.3f}, Proto Acc={proto_mean_acc:.3f}, Fuzzy Acc={fuzzy_mean_acc:.3f} | kNN F1={knn_mean_f1:.3f}, Proto F1={proto_mean_f1:.3f}, Fuzzy F1={fuzzy_mean_f1:.3f} (t={duration:.1f}s)")

    return results

set_seed()
cache = load_cache(CACHE_FILE)
all_results = cache[:] # start from cache

if 'EMBEDDINGS' in globals() and 'LABELS' in globals():
    for name in EMBEDDINGS.keys():
        print(f"\n Running optimized few-shot for dataset: {name}")
        res = run_few_shot_optimized(EMBEDDINGS[name], LABELS[name], dataset_name=name,
                                     few_shot_range=FEW_SHOT_RANGE, n_runs=N_RUNS,
                                     cache_list=all_results, device=None)
        # append and save progressively
        for r in res:
            all_results.append(r)
        save_cache(all_results, CACHE_FILE)

    print("\nDONE — results appended to", CACHE_FILE)
else:
    print("\nEMBEDDINGS and LABELS dictionaries are not defined. Skipping evaluation block.")
    print("Please ensure these global variables are loaded with your data before running.")

# vusial

import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
import json

CACHE_FILE = "results_fewshot_optimized.json"
RESULTS_DIR = Path("./results")
RESULTS_DIR.mkdir(exist_ok=True)

try:
    df_all = pd.read_json(CACHE_FILE)
    if df_all.empty:
        print(f"{CACHE_FILE} is empty — please run Section 6 first.")
    else:
        print(f"Loaded {len(df_all)} results from {CACHE_FILE}")
except Exception as e:
    print(f"Could not read {CACHE_FILE}: {e}")
    df_all = pd.DataFrame()

if not df_all.empty:
    datasets = df_all["Dataset"].unique()
```

```
    for name in datasets:
        df_ds = df_all[df_all["Dataset"] == name].sort_values("Few-shot")
        print(f"\nGenerating graphs {name}")

        # --- Accuracy Plot ---
        plt.figure(figsize=(10, 6))
        plt.plot(df_ds["Few-shot"], df_ds["k-NN Accuracy"], marker='o', label="k-NN")
        plt.plot(df_ds["Few-shot"], df_ds["Prototypical Accuracy"], marker='s', label="Prototypical")
        plt.plot(df_ds["Few-shot"], df_ds["Fuzzy Accuracy"], marker='^', label="Fuzzy Ensemble", linewidth=2.3)

        plt.title(f"{name} — Few-Shot Accuracy")
        plt.xlabel("N-Shot (прикладів на клас)")
        plt.ylabel("Accuracy")
        plt.grid(True, linestyle="--", alpha=0.5)
        plt.legend()
        acc_path = RESULTS_DIR / f"{name}_fewshot_accuracy.png"
        plt.savefig(acc_path)
        plt.show()
        print(f"Saved → {acc_path}")

        # --- F1-Score Plot ---
        plt.figure(figsize=(10, 6))
        plt.plot(df_ds["Few-shot"], df_ds["k-NN F1-Score"], marker='o', label="k-NN F1")
        plt.plot(df_ds["Few-shot"], df_ds["Prototypical F1-Score"], marker='s', label="Prototypical F1")
        plt.plot(df_ds["Few-shot"], df_ds["Fuzzy F1-Score"], marker='^', label="Fuzzy Ensemble F1", linewidth=2.3)

        plt.title(f"{name} — Few-Shot F1-Score")
        plt.xlabel("N-Shot (прикладів на клас)")
        plt.ylabel("F1-Score")
        plt.grid(True, linestyle="--", alpha=0.5)
        plt.legend()
        f1_path = RESULTS_DIR / f"{name}_fewshot_f1_v8.png"
        plt.savefig(f1_path)
        plt.show()
        print(f"Saved → {f1_path}")

    # --- 4. Export summary CSV ---
    summary_cols = [
        "Dataset", "Few-shot",
        "k-NN Accuracy", "Prototypical Accuracy", "Fuzzy Accuracy",
        "k-NN F1-Score", "Prototypical F1-Score", "Fuzzy F1-Score",
        "n_runs", "N_AUG", "AUG_NOISE", "TEMPERATURE", "K_KNN", "Time(s)"
    ]
    csv_path = RESULTS_DIR / "fewshot_summary_v12.csv"
    df_all[summary_cols].to_csv(csv_path, index=False)
    print(f"\nSaved → {csv_path}")

else:
    print("No data — skipping visualization.")
```

Додаток Б

Апробація отриманих результатів

Степаник Олександр Володимерович, здобувач вищої освіти
факультету комп'ютерних інформаційних технологій
Західноукраїнський національний університет, Україна

Науковий керівник: Кіт Іван Романович, викладач кафедри
інформаційно обчислювальних систем і управління
Західноукраїнський національний університет, Україна

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДЛЯ АДОПЦІЇ ТВАРИН

Перенаселення безпритульних собак і котів становить серйозну загрозу для людської спільноти, оскільки вони є переносниками небезпечної вірусної інфекції - сказу. Контроль за чисельністю безпритульних тварин здійснюється через стерилізацію, евтаназію та адопцію. Остання є етичним і популярним методом вирішення проблеми.

Створення автоматизованого вебсайту, який допоможе організаціям, що займаються пошуком нових власників для улюбленців із притулків, активно інформувати та взаємодіяти з користувачами.

Завдання дослідження:

- Аналіз поточної ситуації.
- Дослідження використання штучного інтелекту.
- Розробка дизайну вебсайту.
- Автоматизація вебсайту.
- Додавання розширеного функціоналу.
- Забезпечення взаємодії з користувачами та партнерами

Аналіз та синтез інформації, порівняння існуючих рішень, проектування та розробка, емпіричні методи та методи оцінки ефективності.

Розроблено автоматизований вебсайт із сучасним дизайном та зручним інтерфейсом, який містить систему необов'язкових особистих кабінетів, публікацію оголошень із використанням штучного інтелекту, пошук тварин за зображенням, блог, підтримку для людей із вадами зору, інструкції для користувачів, підказки для нових власників, систему зручної фільтрації, зворотній зв'язок та інформацію про благодійні збори та заходи.

Результати дослідження можуть бути використані організаціями, що займаються адопцією тварин, для підвищення ефективності своєї діяльності та збільшення охоплення цільової аудиторії.

Даний проєкт має потенціал покращити якість обслуговування в інтернет-магазинах шляхом використання інтелектуального аналізу відгуків користувачів та оптимізації бізнес-процесів. Завдяки інтеграції сучасних технологій і штучного інтелекту в основу архітектури, система отримує потужний інструментарій для адаптації до динамічного бізнес-середовища і забезпечення сталого розвитку. Подальше дослідження застосування ШІ відкриває нові можливості для покращення ефективності та конкурентоспроможності в сфері електронної комерції.

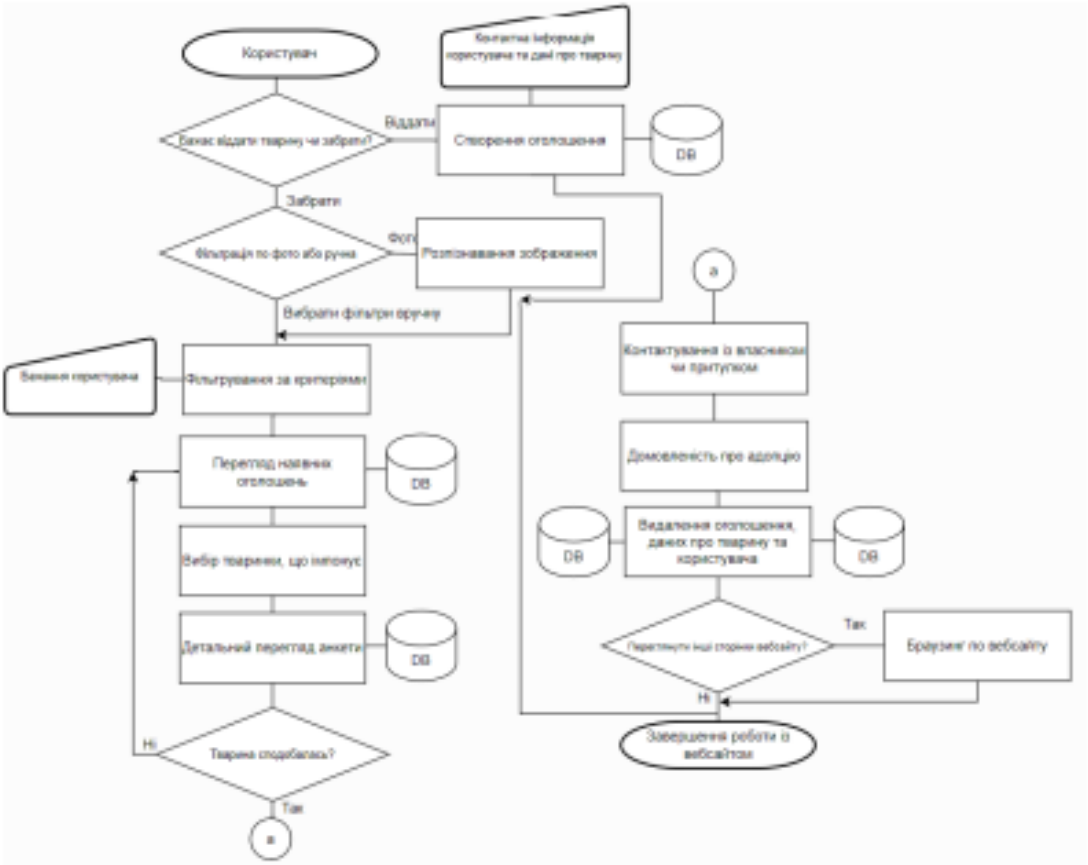

Рис. 1. **Алгоритм роботи вебсайту**

Список використаних джерел:

1. Адопція тварин. [Електронний ресурс] – 2022. URL: https://uk.wikipedia.org/wiki/Адопція_тварин (дата звернення: 22.03.2024).
2. Звідки на вулицях безпритульні тварини? [Електронний ресурс] – 2021. URL: https://epl.org.ua/announces/zvidky-na-vulytsyah-bezprytulni-tvaryny/ (дата звернення: 22.03.2024).
3. Чому домашні тварини опиняються на вулицях та як це виправити. [Електронний ресурс] – 2017. URL: https://dyvys.info/2017/08/19/chotyrylapi-bezhatchenky-chomu-tvaryny-opynyayutsya-na-vulytsi-i-yak-tse-vypravyty/ (дата звернення: 22.03.2024).
4. Ursa.ua [Електронний ресурс] – 2024. URL: https://www.ursaua.com.ua/ (дата звернення: 23.03.2024).

Секція “ШІ у промисловості”

Степанюк Олександр
Студент групи КНм-11
dazerotes@gmail.com
Христина Ліп’янінa-Гончаренко
д.т.н., доцент
kh.lipianina@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

МЕТОД ІДЕНТИФІКАЦІЇ ОЗНАК ТВАРИН НА ОСНОВІ ГЛИБОКО НАВЧАННЯ

Сучасні інформаційні технології все активніше використовуються для вирішення соціально важливих завдань, зокрема — у сфері захисту тварин. Однією з проблем є відсутність централізованих систем ідентифікації тварин, що потребують домівки. У даному дослідженні розглядається підхід до автоматизованого розпізнавання тварин на зображеннях за допомогою методів глибокого навчання. Метою є підвищення ефективності агрегатора оголошень про адопцію через точну ідентифікацію видів, порід і характеристик тварин.

У останні роки глибоке навчання стало ключовим інструментом у задачах ідентифікації тварин. Зокрема, дослідження [1] продемонструвало ефективність використання глибоких згорткових нейронних мереж для розпізнавання видів тварин на зображеннях з екологічних камер-пасток. Модель досягла високої точності класифікації, що свідчить про потенціал CNN у вирішенні подібних задач.

Інше дослідження [2] зосереджено на ідентифікації окремих особин худоби за допомогою глибокого метричного навчання. Використовуючи унікальні візерунки на шкірі тварин, модель змогла точно розпізнавати індивідуальні особини без необхідності носіння ідентифікаційних міток. Цей підхід має значний потенціал для застосування в сільському господарстві та ветеринарії.

Крім того, дослідження [3] представило систему, яка використовує глибоке навчання для ідентифікації домашніх тварин за зображеннями обличчя та тіла. Модель досягла високої точності, що підкреслює можливість застосування таких систем у платформах для адопції тварин.

Запропонована система інтегрує механізми глибокого навчання у вебплатформу, що агрегує оголошення про адопцію тварин. Основу модуля розпізнавання становить модель на базі ResNet34, навчена на відкритих датасетах із фотографіями домашніх тварин (наприклад, Oxford-IIIT Pet Dataset). Модель здійснює класифікацію за видом (кіт/собака/інше) та породою (до 30 категорій).

Передобробка включає зміну розмірів зображення до 224×224, нормалізацію та аугментацію даних для зменшення переобучення. Після

обробки зображення подається на вхід моделі, яка повертає ймовірнісні оцінки для кожної категорії. Архітектура системи наведена на рисунку 1.

Рисунок 1 - Архітектура моделі розпізнавання тварин на зображеннях

Додатково реалізовано модуль рекомендацій, що враховує виведену інформацію для підбору подібних оголошень - на основі ознак, виявлених ШІ. Його приклад можна побачити на рисунку 2.

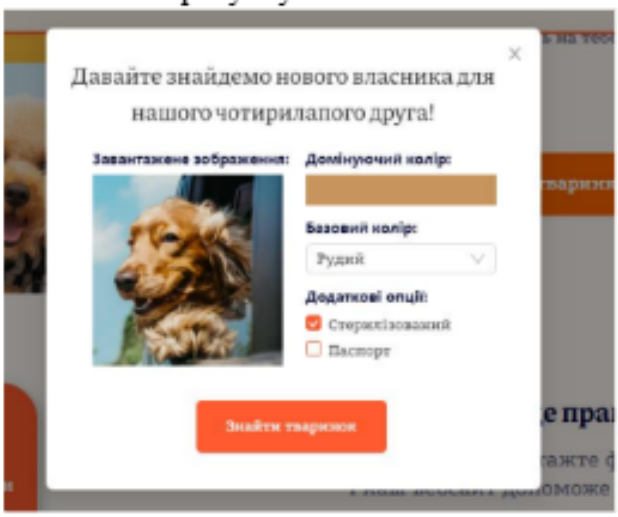

Рисунок 2 - Модуль пошуку схожих оголошень за завантаженим фото тварини

Модель протестовано на вибірці з 1500 зображень, не включених у тренувальний набір. Точність класифікації виду тварини — 97,3%, точність по

породі — 89,5%. Результати по точності для різних моделей наведено в таблиці 1.

Таблиця 1 - Порівняння точності різних моделей у задачі класифікації тварин:

Модель	Точність виду (%)	Точність породи (%)	Час інференсу (мс)
ResNet34	97.3	89.5	42
MobileNetV2	95.1	85.6	23
EfficientNet-B0	96.8	88.2	31

У роботі розроблено метод ідентифікації ознак тварин на основі глибокого навчання, що інтегрований у веб платформу для адопції. Система дозволяє автоматично визначати вид та породу тварини, покращуючи точність пошуку і рекомендацій. Запропонований підхід демонструє потенціал ШІ у цифровізації соціальних ініціатив і має перспективу для масштабування.

Список використаних джерел

1. Islam S. B., Valles D., Hibbitts T. J. та ін. Animal Species Recognition with Deep Convolutional Neural Networks from Ecological Camera Trap Images // Animals. – 2023. – Т. 13, № 9. – С. 1526. – Режим доступу: https://doi.org/10.3390/ani13091526.
2. Andrew W., Gao J., Mullan S. та ін. Visual Identification of Individual Holstein-Friesian Cattle via Deep Metric Learning [Електронний ресурс]. – 2020. – Режим доступу: https://arxiv.org/abs/2006.09205.
3. Azizi E., Zaman L. Deep Learning Pet Identification Using Face and Body // Information. – 2023. – Т. 14, № 5. – С. 278. – Режим доступу: https://doi.org/10.3390/info14050278.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КАФЕДРА ІНФОРМАЦІЙНО-ОБЧИСЛЮВАЛЬНИХ СИСТЕМ І УПРАВЛІННЯ

СЕРТИФІКАТ

Засвідчує, що

Степаник Олександр

взяв(ла) участь у роботі науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (IITAR – 2025)

27-29 травня 2025р.
м. Тернопіль

Голова конференції,
д.т.н., професор

Мирослав КОМАР