Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ НЕЙРОМЕРЕЖ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-23-2
Яковенко В.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Яковенку Віктору Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження використання нейромереж для класифікації зображень

затверджена наказом по університету від 25 листопада 2024 року № 1246Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 грудня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мова програмування Python, середовище розробки PyCharm.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз методів класифікації зображень за допомогою нейромереж.
2. Методи класифікації зображень нейромережами.
3. Програмна реалізація нейромережі для класифікації зображень.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми класифікації зображень, постановка задачі, об'єкт дослідження, мета роботи, сучасні методи класифікації зображень, трансформер зору, тестові зображення, апробація тез, наукова новизна.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	25.11.2024	
2	Аналіз завдання, підбір літератури	26.11.24-28.11.24	
3	Аналіз літератури з досліджуваної проблеми	29.11.24-30.11.24	
4	Аналіз технічних засобів	01.12.24-03.12.24	
5	Розробка методу	04.12.24-06.12.24	
6	Програмна реалізація	07.12.24-08.12.24	
7	Оформлення пояснювальної записки	09.12.24-10.12.24	
8	Перевірка на плагіат	12.12.2024	
9	Рецензування	14.12.2024	
10	Підготовка презентації та доповіді	15.12.2024	
11	Занесення роботи в електронний архів	04.01.2025	
12	Попередній захист кваліфікаційної роботи	07.01.2025	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 75 с., 1 табл., 40 рис., 1 дод., 40 джерел.

НЕЙРОННІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ, КОМП'ЮТЕРНИЙ ЗІР, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, VISION TRANSFORMER, DISCORD БОТ.

Об'єктом дослідження є метод класифікації зображень за допомогою нейронних мереж, зокрема його застосування в сучасних технологічних умовах.

Метою дослідження є розробка, реалізація та дослідження методу, що базується на використанні нейронних мереж для класифікації зображень, які здатні ефективно виділяти ознаки.

Використано методи машинного навчання. Проведено дослідження методів класифікації зображень, а також аналіз ефективності застосування трансформера зору для вирішення задач класифікації. Розглянуто особливості моделювання та оптимізації архітектури нейромережі, розроблено алгоритм попередньої обробки та класифікації зображень.

У результаті дослідження здійснено програмну реалізацію застосунку для класифікації зображень, який забезпечує високу точність та ефективність роботи.

NEURON NETWORKS, ARTIFICIAL INTELLIGENCE, COMPUTER VISION, IMAGE CLASSIFICATION, VISION TRANSFORMER, DISCORD BOT.

The object of the research is the method of image classification using neural networks, particularly its application in modern technological conditions.

The aim of the research is to develop, implement and investigate a method based on the use of neural networks for image classification, which can effectively distinguish features.

Machine learning methods were used. A study of methods of image classification was carried out, as well as an analysis of the effectiveness of the use of a vision transformer for solving classification problems. The peculiarities of modeling and optimization of neural network architecture are considered, and an algorithm for image preprocessing and classification is developed.

As a result of the research, the software implementation of the image classification application, which ensures high accuracy and efficiency of work, was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

CV – Computer Vision (комп’ютерний зір)

CNN – Convolutional Neural Networks (згорткові нейронні мережі)

ViT – Vision Transformer (зоровий трансформер)

GPT – Generative Pre-trained Transformer (породжувальний попередньо тренований трансформер)

MLP – Multilayer Perceptron (багатошарового персептрон)

RNN – Recurrent Neural Networks (рекурентні нейронні мережі)

YOLO – You Only Look Once (дивишся лише раз)

SSD – Single Shot MultiBox Detector (детектор розпізнавання за один прохід МультиБокс)

IDE – Integrated Development Environment (інтегроване середовище розробки)

CUDA – Compute Unified Device Architecture (архітектура обчислювального уніфікованого пристрою)

GPU – Graphics Processing Unit (графічний процесор)

API – Application Programming Interface (інтерфейс програмування застосунків)

ВСТУП

Нейронні мережі є одним із напрямків машинного навчання, який виник ще в 1940-х роках. Цей підхід ґрунтується на біологічних нейронних мережах тварин, які складаються з нейронів головного та спинного мозку центральної нервової системи. Звісно, штучні нейронні мережі є значно спрощеною версією біологічних, і їхня мета полягає в розв'язанні завдань, які людський мозок виконує щодня.

В умовах стрімкого розвитку технологій, їх автоматизації, поширення робототехніки, розвитку інтернет-речей та систем ШІ, одними з найбільш актуальних напрямів залишаються машинне навчання, комп'ютерний зір та розпізнавання об'єктів на зображеннях [1-5].

Нейронні мережі зробили революцію в області комп'ютерного зору, дозволивши машинам розпізнавати та аналізувати зображення. Вони стають все більш популярними завдяки своїй здатності вивчати складні моделі та особливості. Особливо CNN є найпопулярнішим типом нейронних мереж, які використовуються в обробці зображень. Але також ViT стають все більш популярними останнім часом завдяки революційним досягненням GPT та інших архітектур на основі трансформаторів у обробці природної мови [6]. Загалом, нейронні мережі обробляють і розпізнають зображення різними способами. Це залежить від архітектури мережі та проблеми.

Нейронні мережі відіграють ключову роль у класифікації зображень, оскільки вони здатні автоматично навчатися розпізнавати візуальні патерни в даних [7]. Використовуючи глибоке навчання, нейронні мережі можуть аналізувати величезні масиви зображень і класифікувати їх на основі різних ознак, таких як форми, кольори або текстури. Цей підхід дозволяє досягти високої точності в задачах класифікації зображень, що робить нейронні мережі незамінними для таких застосувань, як розпізнавання об'єктів, аналіз медичних зображень або автоматизація процесів в різних галузях.

Нейронні мережі використовують для задач, для яких побудувати алгоритм вирішення – складна та тривала для людини задача. Приклади таких задач:

– автоматизація контролю якості продукції;

– медична діагностика;

– ідентифікація підозрілих осіб у системах відеоспостереження;

– розпізнавання дорожніх знаків, пішоходів та інших автомобілів;

– класифікація зображень для автоматичного тегування фотографій, покращення рекомендацій та створення контенту в соціальних мережах;

– розпізнавання облич.

Актуальність дослідження полягає у зростаючій потребі в ефективних методах обробки та аналізу зображень у сучасному світі. Протягом теперішнього десятиліття обсяги даних, що генеруються щоденно, зростають із неймовірною швидкістю. Соціальні мережі, системи відеоспостереження, медичні пристрої – всі вони продукують величезні масиви зображень. Ефективні методи класифікації, зокрема ті, що ґрунтуються на нейронних мережах, дозволяють не лише обробляти ці дані, але й витягувати з них корисну інформацію.

1 АНАЛІЗ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖ

1.1 Поняття нейромережі

Нейронна мережа – це програма або модель машинного навчання, яка приймає рішення подібно до людського мозку, використовуючи процеси, які імітують те, як біологічні нейрони працюють разом, щоб ідентифікувати явища, зважувати варіанти та робити висновки.

Кожна нейронна мережа складається з шарів вузлів або штучних нейронів – вхідного рівня, одного або кількох прихованих шарів і вихідного рівня. Нейронна мережа вважається неглибокою, якщо має лише один прихований шар (рис. 1.1), і глибокою, якщо кількість прихованих шарів більша за один (рис. 1.2). Шари містять в собі вузли, які є штучними нейронами. Кожен вузол підключається до інших і має власну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі.

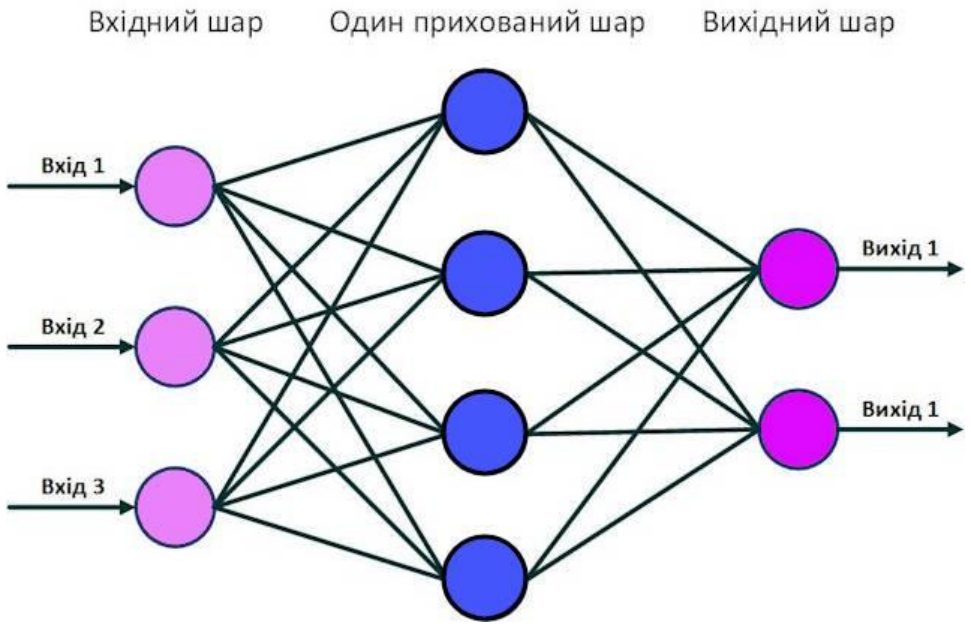

Рисунок 1.1 – Неглибока нейронна мережа

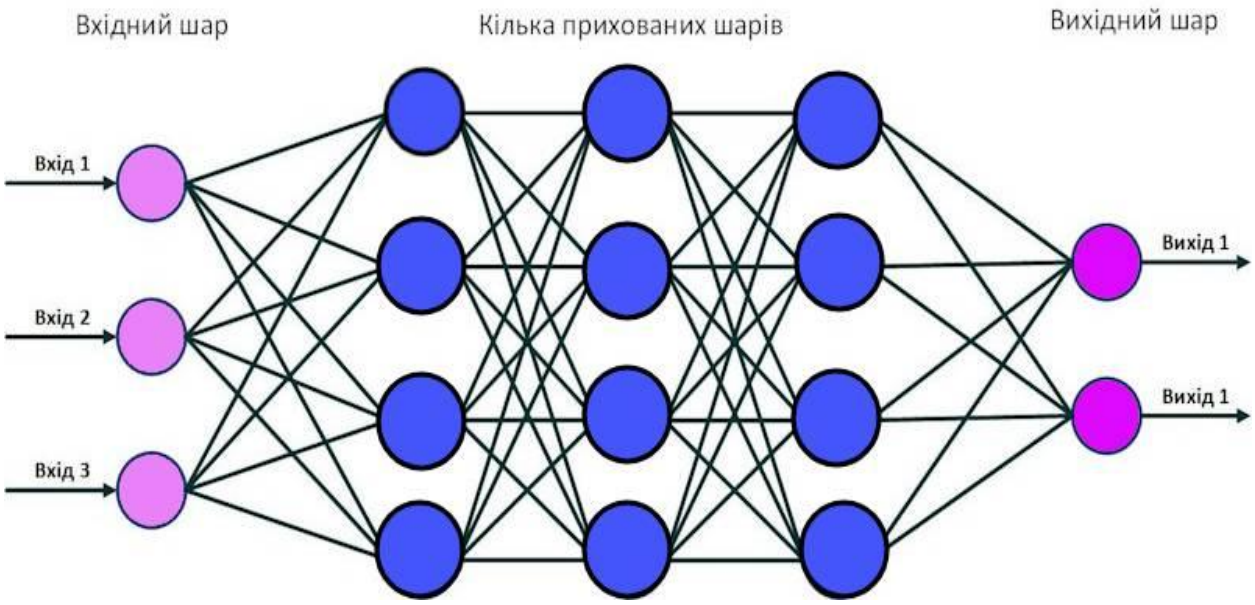

Рисунок 1.2 – Глибока нейронна мережа

Нейронні мережі покладаються на навчальні дані, щоб навчатися та підвищувати свою точність з часом. Після їх точного налаштування на точність вони стають потужними інструментами в інформатиці та штучному інтелекті, що дозволяють класифікувати та кластеризувати дані з високою швидкістю. Завдання з розпізнавання мовлення або розпізнавання зображень можуть тривати хвилини чи години порівняно з ручною ідентифікацією експертів-людей. Одним із найвідоміших прикладів нейронної мережі є пошуковий алгоритм Google.

Нейронні мережі іноді називають штучними нейронними мережами або імітованими нейронними мережами. Вони є підмножиною машинного навчання та є основою моделей глибокого навчання.

У сучасній практиці глибокі нейронні мережі часто використовують складніші архітектури, такі як CNN або RNN [8], які набагато краще підходять для складних задач. CNN спеціалізуються на обробці даних із просторовою структурою, таких як зображення або відео, де вони можуть автоматично виділяти просторові ознаки на різних рівнях. RNN, у свою чергу, відмінно працюють із послідовними даними, такими як текст, аудіо або часові ряди, завдяки здатності враховувати контекст попередніх елементів у послідовності.

Вони широко застосовуються в задачах обробки письмової та усної мови, генерації тексту, аналізу фінансових даних, розпізнавання мови та багатьох інших сферах, де важливий порядок даних або зв’язок між їх елементами.

1.2 Згорткові нейронні мережі

CNN є спеціальним типом нейронних мереж, який чудово підходить для роботи з зображеннями [9, 10]. Їх широко використовують у задачах комп’ютерного зору, таких як генерація й класифікація зображень, розпізнавання об’єктів і поз. Раніше ці проблеми намагалися вирішувати за допомогою класичних нейронних мереж, зокрема MLP (найпростіший варіант глибокої нейронної мережі), а також різними евристичними підходами, але CNN значно покращили результати в цій сфері.

Крім обробки двовимірних зображень, CNN також здатні працювати з одновимірними та тривимірними сигналами.

Наприклад, одновимірними даними можуть бути аудіосигнали, зібрані мікрофоном, які згорткові нейронні мережі використовують для задач розпізнавання мови (рис. 1.3).

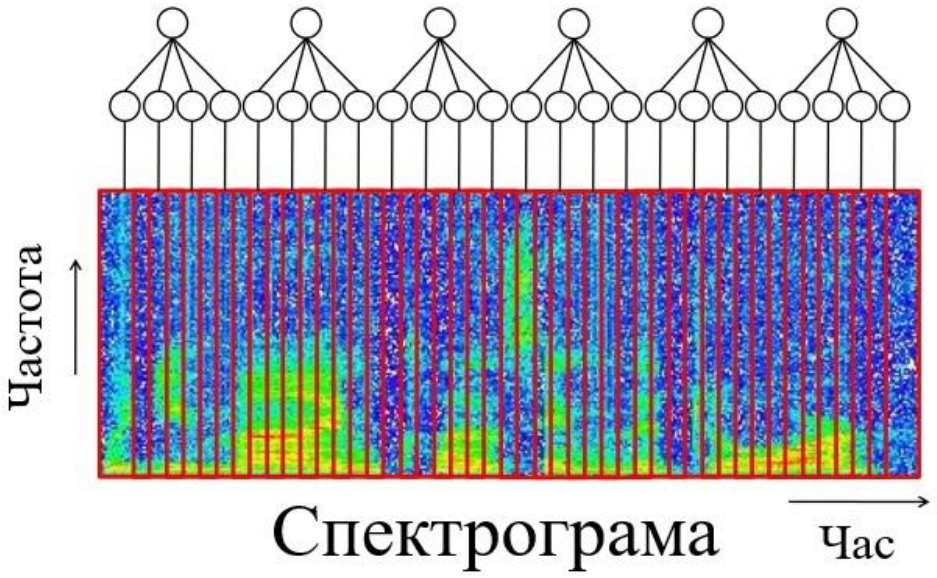

Рисунок 1.3 – Згорткова глибока нейронна мережа для звукового сигналу

Що ж до тривимірних даних, то ними є відео [11], де третім виміром є кількість кадрів в секунду, і CNN можуть аналізувати такі дані для розпізнавання об’єктів або дій у динаміці (рис. 1.4).

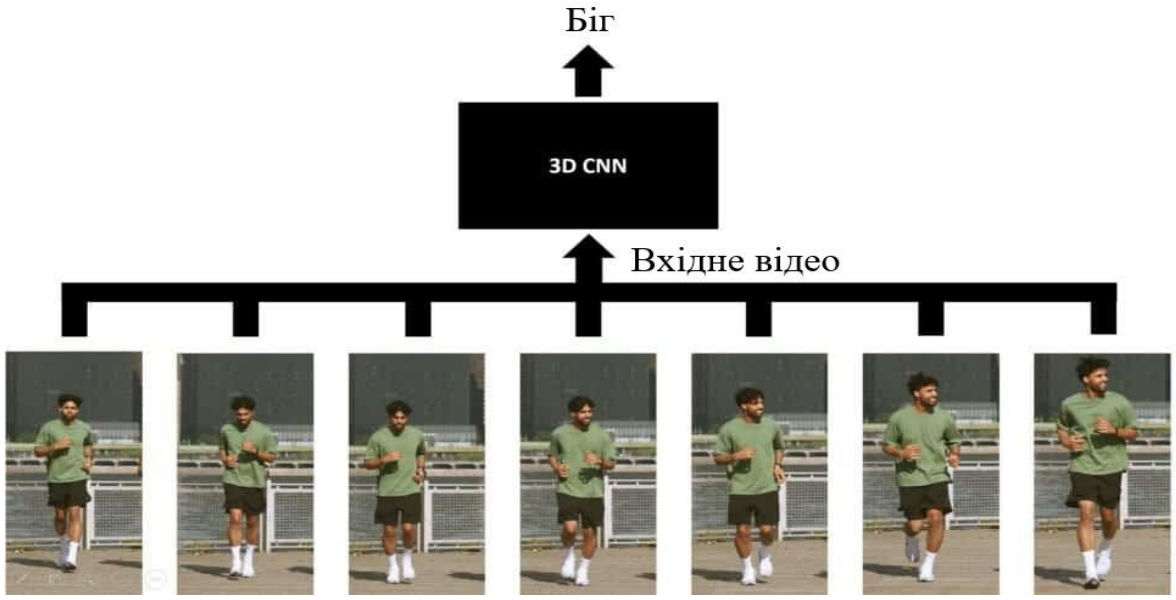

Рисунок 1.4 – Приклад розпізнавання бігу за допомогою CNN

1.2.1 Операція згортки

Згорткові нейронні мережі отримали свою назву через процес виявлення країв, де менші локальні ознаки зображення об’єднуються в більш загальні. Цей процес математично описується операцією, відомою як згортка, що виглядає наступним чином:

$$y[x,y] = f[x,y] * g[x,y] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot g[x-i, y-j], \qquad (1.1)$$

де $y[x,y]$ – значення вихідного сигналу;

$f[x,y]$ – вхідний сигнал;

$g[x,y]$ – ядро згортки, яке застосовується до вхідного сигналу.

Операція згортки в контексті обробки зображень включає поелементне множення частини вхідного зображення на ядро з подальшим сумуванням результатів. Це дає новий піксель у вихідному зображенні. Застосування згортки до зображення показано на рисунку 1.5.

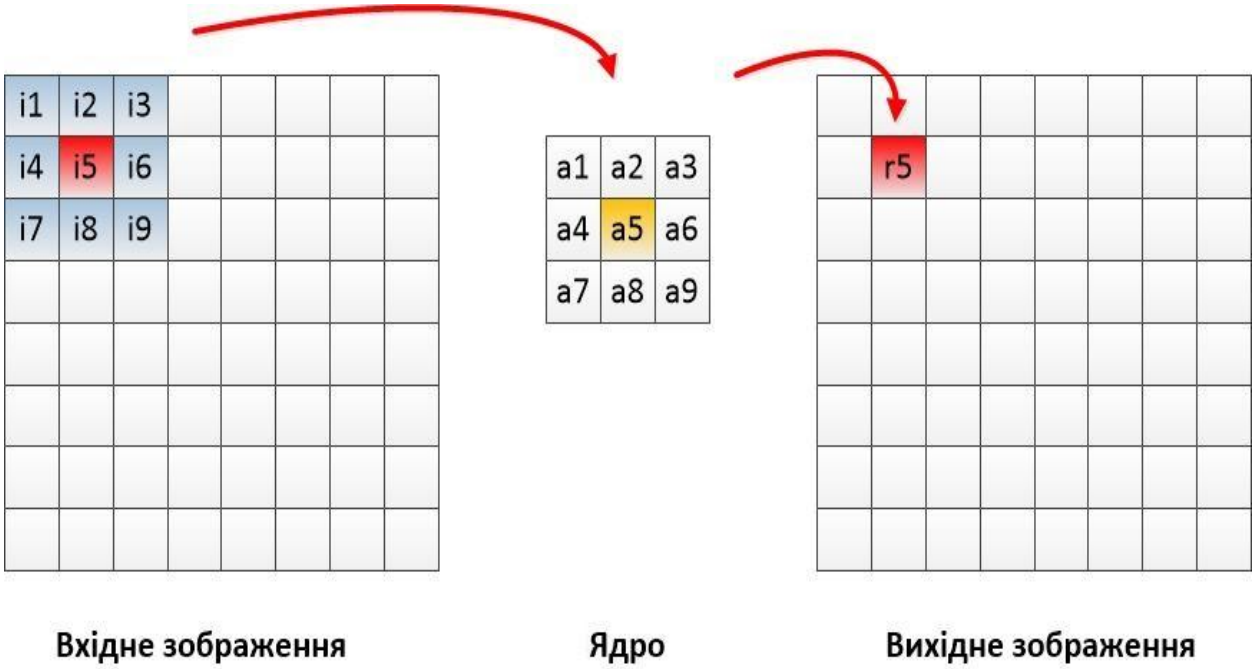

Рисунок 1.5 – Операція згортки зображення

1.2.2 Пулінг

Пулінг – це операція, яка використовується в CNN для зменшення розмірів вхідних даних, зберігаючи при цьому важливі характеристики. Це допомагає зменшити обчислювальну складність моделі та уникнути перенавчання [12].

Існує три основних види пулінгу:

– Max Pooling вибирає з кожного блоку вхідних даних максимальне значення. Ця процедура дозволяє зберігати найсильніші ознаки зображення. Наприклад, якщо є блок розміром 2×2, Max Pooling вибере найбільше значення в цьому блоці. Процедуру Max Pooling показано на рисунку 1.6;

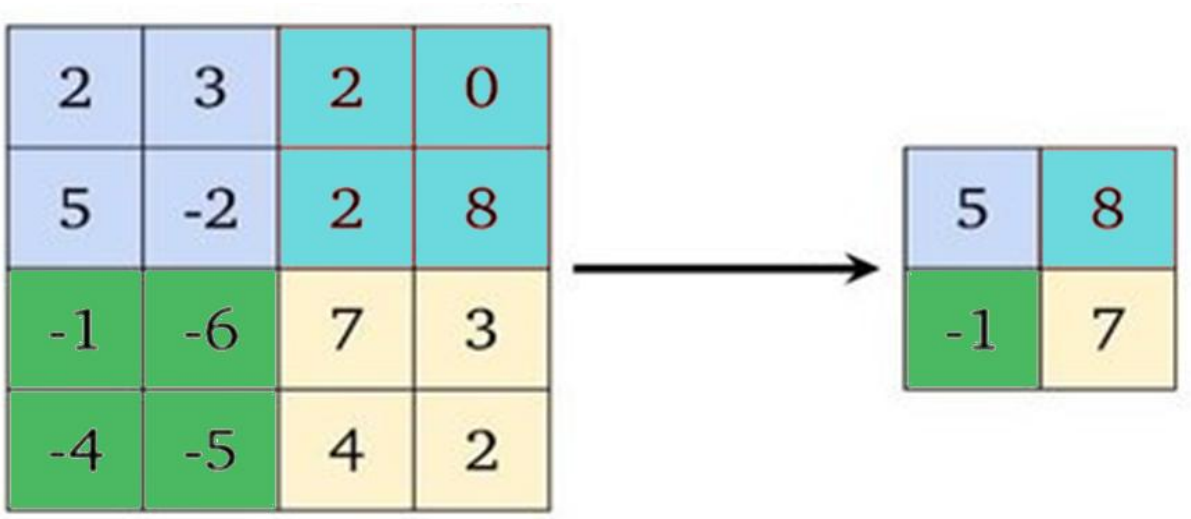

Рисунок 1.6 – Принцип роботи Max Pooling

– Average Pooling обчислює з кожного блоку вхідних даних середнє значення. Цей підхід згладжує результати і може бути корисним у випадках, коли необхідно зберегти загальний контекст ознак. Результат роботи Average Pooling показано на рисунку 1.7;

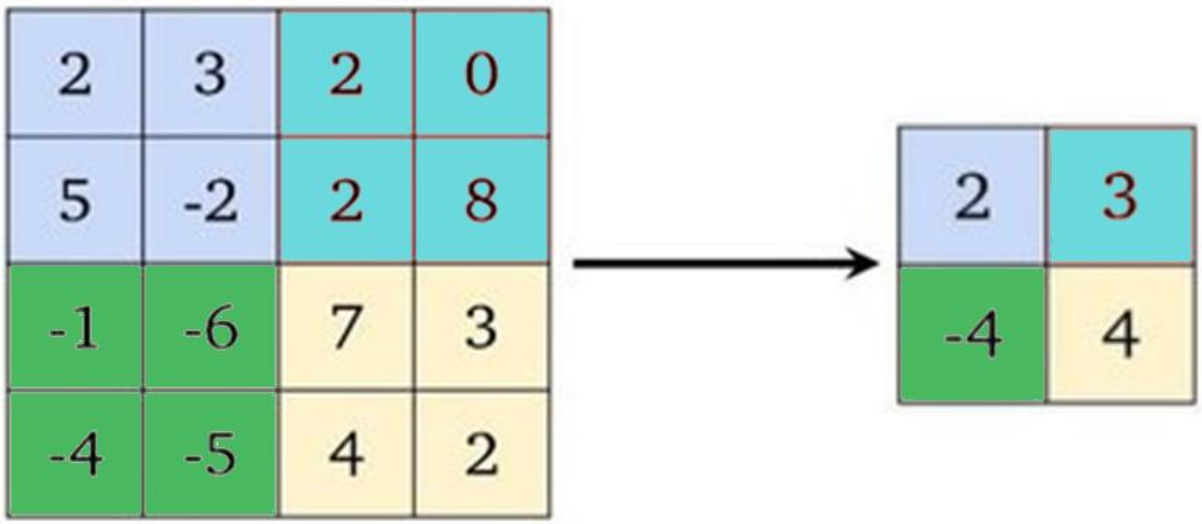

Рисунок 1.7 – Принцип роботи Average Pooling

– Global Pooling зменшує всю карту ознак до одного значення на кожний канал. Наприклад, глобальний Max Pooling (рис. 1.8) або глобальний Average Pooling. Це часто використовується перед повнозв’язними шарами в CNN.

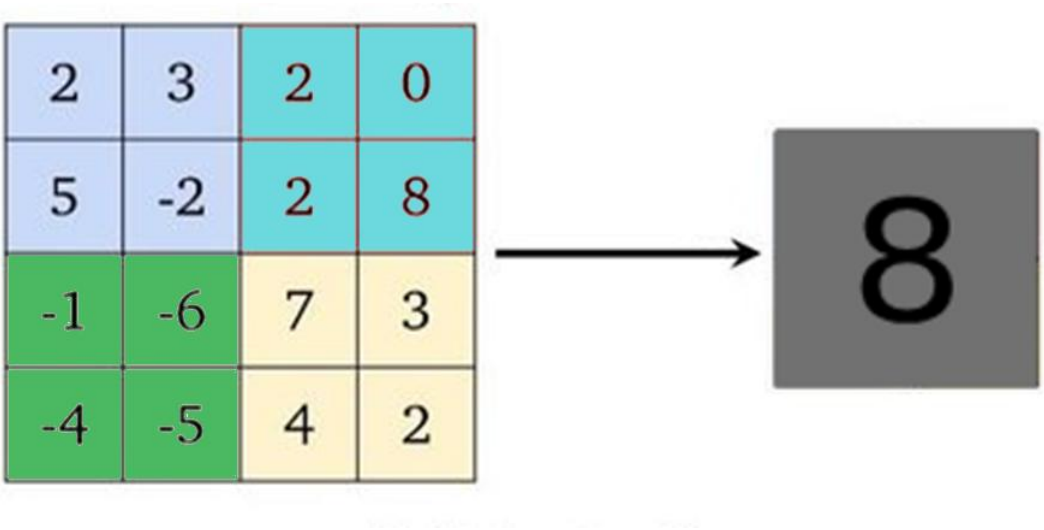

Рисунок 1.8 – Приклад Global Max Pooling

Пулінг є критично важливою операцією в CNN. Основними перевагами пулінгу є те, що він дозволяє зменшити кількість параметрів і обчислень. Це зменшує обчислювальні витрати і потребу в пам’яті, прискорюючи роботу моделі. Також пулінг допомагає зробити мережу менш чутливою до дрібних зсувів або змін вхідних даних і скорочує кількість ознак, що знижує ризик перенавчання моделі. Усі ці переваги є важливими для завдань комп’ютерного зору, де об’єкти можуть з’являтися в різних положеннях або масштабах. Використання пулінгу, зокрема максимального, також забезпечує збереження важливих інформаційних елементів, що може покращити точність класифікації в задачах комп’ютерного зору.

1.3 RNN

RNN зазвичай використовуються для обробки послідовних даних, таких як текст або часові ряди, їх також можна застосувати для класифікації зображень, при умові якщо розглядати зображення як послідовність пікселів або блоків пікселів. На відміну від CNN, RNN здатні запам’ятовувати ключові деталі про отримані вхідні дані завдяки своїй внутрішній пам’яті (рис. 1.9), що дозволяє їм з високою точністю передбачати майбутні події [13].

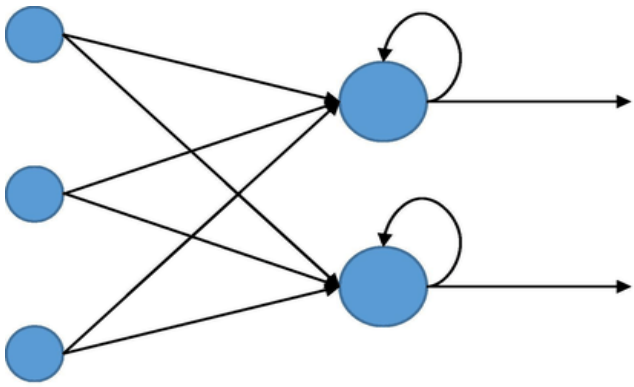

Рисунок 1.9 – Рекурентна нейромережа

RNN не є стандартним вибором для класифікації зображень, вони можуть бути корисними в спеціальних випадках або в поєднанні з CNN. Найчастіше використання RNN виправдано, коли потрібно враховувати послідовну структуру у зображеннях, наприклад, у задачах розпізнавання рукописного тексту або класифікації зображень з певною послідовністю елементів.

Основна ідея полягає в тому, щоб розглядати зображення як двовимірну сітку пікселів і перетворювати її в послідовність. Наприклад, можна «прочитати» зображення піксель за пікселем, рядок за рядком або блок за блоком. Кожен рядок або блок пікселів можна розглядати як елемент послідовності, який передається на вхід RNN.

Кожен крок RNN обробляє черговий елемент послідовності і передає стан на наступний крок, поступово «запам’ятовуючи» контекст зображення. Після обробки всієї послідовності фінальний стан RNN можна використовувати для класифікації зображення.

Найбільш ефективним підходом є поєднання CNN з RNN. CNN спочатку використовується для вилучення просторових ознак зображення, обробляючи локальні патерни за допомогою згорток. Це допомагає зберегти ключову інформацію про форму, текстуру та інші характеристики зображення. Потім ці ознаки можуть подаватися в RNN для обробки вже як послідовності високорівневих ознак. Таким чином, RNN допомагає враховувати глобальний контекст або залежності між різними частинами зображення.

1.4 Класифікація зображень нейронними мережами

За останні роки задача класифікації зображень стала дуже популярною завдяки зростанню кількості проєктів, спрямованих на автоматизацію керування транспортними засобами, розпізнавання облич, автомобільних номерів та інших об’єктів за допомогою камер. Такі технології активно застосовуються у різних сферах нашого життя. Класифікація полягає в ідентифікації об’єкта на зображенні та віднесенні його до певної категорії (рис. 1.10).

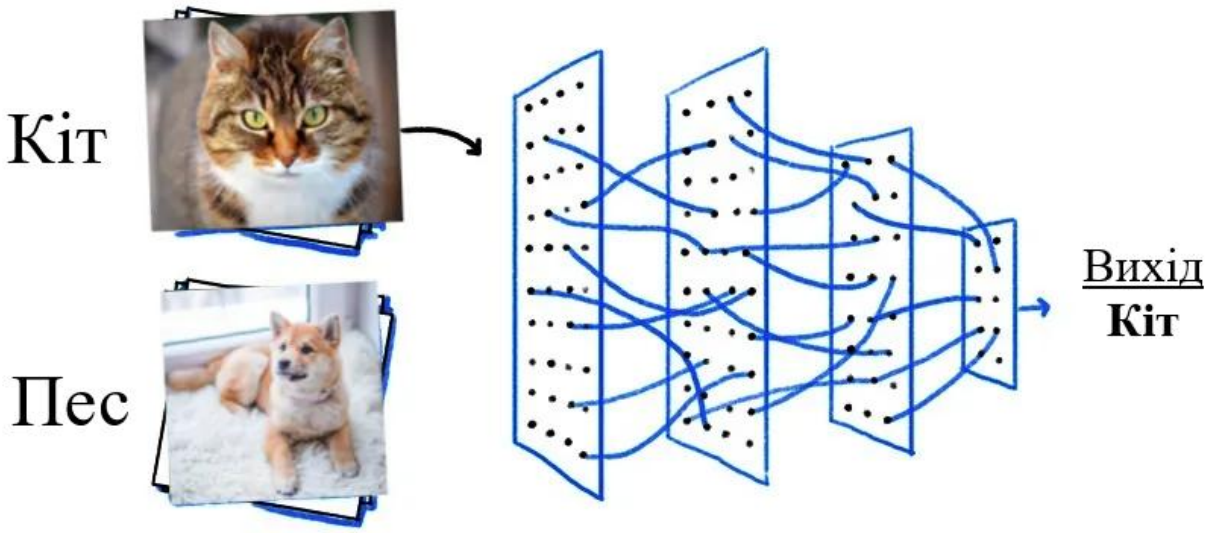

Рисунок 1.10 – Приклад класифікації зображень

Алгоритми машинного навчання, що використовуються для класифікації зображень, постійно вдосконалюються експертами, однак загальна структура їх виконання залишається незмінною. Процес класифікації зазвичай виконується покроково:

Крок 1. Нормалізація даних. Це початковий етап, де зображення готують до подальшої обробки. Включає, наприклад, масштабування або зменшення шуму, щоб забезпечити рівні умови для роботи алгоритму.

Крок 2. Виділення та сегментація об’єкта. На цьому етапі на зображенні виділяють ту частину, де знаходиться об’єкт, який необхідно класифікувати. Це може бути як простий контур, так і складна форма.

Крок 3. Виділення ключових ознак. Далі визначаються найважливіші характеристики об’єкта, такі як форма, текстура, колір тощо. Такі ознаки дозволять алгоритму «зрозуміти», з чим він має справу.

Крок 4. Класифікація. Нарешті, на основі виділених ознак, об’єкт порівнюється з відомими класами, і йому присвоюється певна категорія.

Цей базовий процес може мати додаткові кроки, такі як аугментація даних – це штучне збільшення кількості навчальних зразків для підвищення точності моделі. Також, якщо використовуються сучасні глибокі нейронні мережі, вони можуть автоматично виділяти важливі ознаки, що значно спрощує процес і підвищує точність при роботі зі складними зображеннями.

1.5 Навчання нейронних мереж

У ході навчання нейронної мережі модель штучного інтелекту вчиться виконувати певні завдання з урахуванням наданих їй даних.

Головне завдання навчання нейромереж полягає у розвитку у них здатності вирішувати поставлені завдання. Деякими з цих завдань можуть бути:

– автоматизація процесів;

– прогнозування;

– обробка великих масивів даних;

– покращення якості прийняття рішень;

– регресія;

– кластеризація.

Є множина методів «тренування» нейронних мереж, але вони ґрунтуються на двох ключових принципах: за допомогою вчителя і без нього.

Це відбувається так само, як і в людини: можна набувати нових знань під керівництвом наставника, який підкаже та скоригує окремі моменти, а можна займатися самоосвітою. В останньому випадку людина спирається лише на свій особистий досвід та спостереження.

1.5.1 Навчання з учителем

Навчання з учителем – це найбільш поширений підхід, при якому нейромережа навчається на основі попередньо розмічених даних. Ці дані включають вхідні значення і відповідні їм цільові. Нейронна мережа навчається передбачати останні на основі вхідних даних [14].

Процес навчання відбувається наступним чином. Спочатку потрібно підготувати набір вхідних даних, для яких відомі очікувані результати. Наприклад, якщо ми хочемо навчити нейронну мережу розпізнавати зображення кішок, нам потрібно підготувати набір зображень кішок (вхідні дані) та міток, які вказують, що на цих зображеннях дійсно зображені кішки (очікувані результати).

На етапі процесу навчання нейронна мережа переглядає вхідні дані та робить передбачення на основі своїх поточних параметрів (ваг). Потім ці прогнози порівнюються з очікуваними результатами.

Якщо прогноз нейронної мережі не відповідає очікуваному результату, запускається процес, відомий як зворотне поширення помилки. У цьому процесі визначається, як кожна вага в нейромережі вплинула на загальну помилку. Потім ці ваги коригуються в напрямку, що дозволяє зменшити цю помилку.

Цей процес повторюється багато разів, використовуючи нові дані з навчального набору. З часом ваги нейронної мережі налаштовуються так, щоб мінімізувати розбіжність між прогнозами та очікуваними результатами.

Далі навчені нейронні мережі перевіряються на нових даних, які раніше не використовувалися в процесі навчання. Це дає змогу оцінити, наскільки ефективно модель може застосовувати свої знання в нових ситуаціях.

Важливо, що навчання з викладачем вимагає великої кількості розмічених даних, що може бути дорогим та трудомістким процесом. Для досягнення хороших результатів навчання важливо мати якісні та точні мітки. Неправильні або неякісні дані можуть призвести до навчання моделі з помилками, що знижує її ефективність. Забезпечення високої якості даних вимагає додаткових ресурсів і зусиль.

1.5.2 Навчання без учителя

Навчання без учителя – це процес, під час якого нейронна мережа оперує нерозміченими даними. Головна мета полягає у виявленні прихованих закономірностей, структур або взаємозв’язків у даних без будь-якої попередньої інформації про очікувані результати.

Процес починається з підготовки даних. На відміну від навчання з учителем, де дані повинні бути міченими, тут розмітка не потрібна. Достатньо мати лише набір вхідних показників. Наприклад, для навчання нейронної мережі групувати зображення котів і собак достатньо просто зібрати їх зображення, не вказуючи, де коти, а де собаки.

Після цього нейронна мережа приступає до навчання, намагаючись виявити структуру або закономірності в цих даних. Це можуть бути процеси кластеризації, коли подібні об’єкти групуються разом, або пошук аномалій, де виділяються об’єкти, що суттєво відрізняються від інших. Також можливо виконання зменшення розмірності – скорочення кількості ознак без суттєвої втрати інформації.

Протягом навчання ваги нейронної мережі постійно коригуються, щоб точніше відображати структуру даних. Процес проходить кілька ітерацій до

того моменту, поки мережа не налаштується на оптимальне відображення знайдених закономірностей. Завершальним етапом є тестування навченої моделі на нових, раніше невикористаних даних для оцінки того, наскільки ефективно вона виявляє структури в нових ситуаціях.

1.6 Постановка задачі дослідження

Методи та алгоритми класифікації зображень залишаються актуальними, оскільки допомагають вирішувати реальні задачі в різних сферах, де активно використовуються зображення – таких як медицина, технології, мистецтво, наука та інші. Спираючись на це, важливо дослідити застосування нейронних мереж для класифікації зображень з урахуванням можливостей сучасних технологій.

Об’єктом дослідження є метод класифікації зображень за допомогою нейронних мереж, зокрема його застосування в сучасних технологічних умовах.

Метою дослідження є розробка, реалізація та дослідження методу, що базується на використанні нейронних мереж для класифікації зображень, які здатні ефективно виділяти ознаки.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів класифікації зображень за допомогою нейронних мереж;

– дослідити підходи для виділення ключових ознак зображень за допомогою нейромереж;

– розробити метод класифікації зображень на основі нейронних мереж із застосуванням сучасних методів обробки даних;

– створити комп’ютерну модель для автоматизованого процесу класифікації зображень та аналізу результатів;

– провести порівняння запропонованого методу з існуючими підходами.

2 МЕТОДИ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ НЕЙРОМЕРЕЖАМИ

2.1 Основні підходи класифікації зображень

Існує два основних підходи класифікації зображень: класифікація на основі пікселів та класифікація на основі об’єктів.

Пікселі є базовими елементами зображення і класифікація на основі аналізу пікселів є основним методом. Існують алгоритми, що або використовують тільки спектральну інформацію окремих пікселів для класифікації, або поєднують спектральну інформацію з просторовою (дані про прилеглі пікселі). Методи класифікації на рівні пікселів враховують лише яскравість кожного пікселя, тоді як об’єктна класифікація використовує як спектральну, так і просторову інформацію.

До методів класифікації на основі пікселів належать такі, як метод мінімальної відстані до середнього, максимальна ймовірність та метод відстані Махаланобіса. Вони використовують середні значення та дисперсії для різних класів і працюють за принципом вимірювання «відстані» між цими середніми значеннями та значеннями пікселів. Методи класифікації на основі пікселів мають обмеження, оскільки не враховують інформацію з сусідніх пікселів, але мають перевагу в простоті реалізації та ефективності у задачах з базовими візуальними характеристиками.

Натомість класифікація на рівні об’єктів враховує і спектральні, і просторові дані, оскільки включає прилеглі пікселі в процес класифікації. Під «об’єктом» розуміють не конкретний об’єкт на зображенні, а суміжну область пікселів. Цей підхід базується на розпізнаванні об’єктів або груп пікселів, які утворюють певні смислові одиниці на зображенні. Він дозволяє враховувати не тільки самі пікселі, але й їх розташування та взаємозв’язки між ними.

Таким чином, класифікація на основі пікселів більше підходить для простих задач, тоді як класифікація на основі об’єктів є більш універсальною та потужною для складних візуальних задач.

2.2 Попередня обробка даних

Сучасні та надійні системи класифікації зображень здебільшого використовують підхід, орієнтований на об’єкти, де зображення потрібно відповідним чином підготувати. Спочатку необхідно вибрати та попередньо обробити об’єкти або області на зображенні [15].

Перед тим як класифікувати зображення чи його окремі об’єкти, комп’ютер повинен «зрозуміти» дані, які воно містить. Це досягається шляхом попередньої обробки та підготовки зображень для подальшого введення в класифікаційний алгоритм, використовуючи техніки виявлення об’єктів [16]. Цей процес є важливим етапом підготовки зображень і даних для навчання моделей машинного навчання.

Попередня обробка зображень є важливою для застосунків комп’ютерного зору, оскільки вона покращує точність до 30% [17]. Наступні етапи попередньої обробки відіграють дуже важливу роль в оптимізації зображень для аналізу, оскільки покращують вхідні дані для моделей машинного навчання:

– зменшення шуму;

– підвищення контрастності;

– зміна розміру;

– нормалізація;

– аугментація;

– перетворення кольорових просторів (наприклад, з RGB у відтінки сірого або HSV);

– виділення контурів;

– підвищення різкості;

– видалення небажаних артефактів.

Правильна попередня обробка зображень не тільки підвищує точність, але й знижує обчислювальну складність, полегшуючи моделі навчання важливим ознакам і покращуючи їх роботу в реальних застосунках.

2.2.1 Локалізація зображення

Виявлення об’єктів здійснюється за допомогою різних методів. Спосіб попередньої обробки залежить від того, чи на зображенні присутній один об’єкт інтересу, чи їх кілька. Якщо на зображенні є лише один об’єкт, застосовується техніка локалізації зображення [18]. Пікселі зображення мають числові значення, що інтерпретуються комп’ютером для відтворення відповідних кольорів і відтінків. Навколо об’єкта, що цікавить, створюється обмежувальна рамка, яка вказує комп’ютеру, яка частина зображення є важливою та які піксельні значення її описують.

Локалізація є важливим етапом у процесі розпізнавання об’єктів, оскільки вона допомагає не тільки визначити наявність об’єкта, але й точно вказати його місцезнаходження на зображенні. Обмежувальна рамка дозволяє моделі фокусуватися на конкретній частині зображення, що спрощує подальшу класифікацію (рис. 2.1). У багатьох випадках локалізація може бути доповнена додатковими параметрами, наприклад, оцінкою впевненості в тому, що об’єкт дійсно є в межах обмежувальної рамки. Ця інформація особливо важлива в складних або завантажених сценах, де можуть бути кілька схожих об’єктів або де частини об’єктів можуть бути частково закриті іншими елементами зображення.

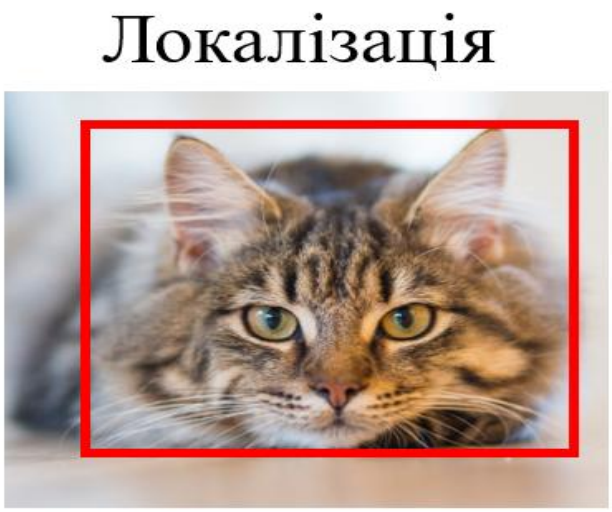

Рисунок 2.1 – Локалізація зображення

2.2.2 Виявлення об’єктів

Якщо ж на зображенні декілька об’єктів, застосовується метод виявлення об’єктів (рис. 2.2), щоб визначити обмежувальні рамки для кожного об’єкта [19]. Цей метод дозволяє одночасно ідентифікувати та локалізувати кілька об’єктів на одному зображенні, що є важливим аспектом у багатьох застосуваннях комп’ютерного зору. Виявлення об’єктів може виконуватися за допомогою різних алгоритмів, таких як YOLO, SSD та Faster R-CNN. Кожен з цих алгоритмів має свої переваги та недоліки, проте всі вони намагаються оптимізувати процес виявлення, щоб зменшити час обробки та підвищити точність. Наприклад, YOLO обробляє зображення за один прохід, що дозволяє досягти високої швидкості в реальному часі, тоді як Faster R-CNN використовує попередньо обчислені регіональні пропозиції, що забезпечує високу точність, але вимагає більше часу на обробку. Після виявлення об’єктів, для кожного з них створюються обмежувальні рамки, які вказують на їх точне розташування.

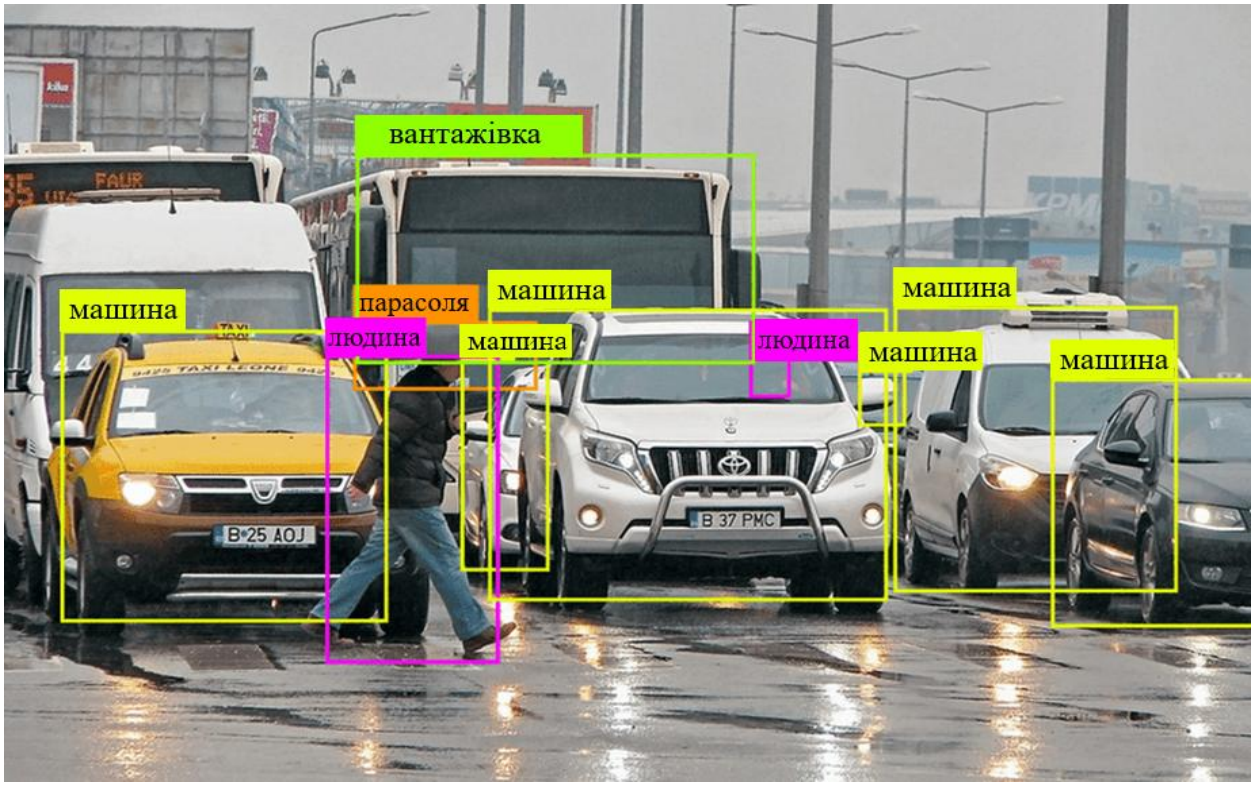

Рисунок 2.2 – Виявлення об’єктів на зображенні

2.3 Архітектура згорткових нейронних мереж

Класична архітектура CNN складається з різних шарів, кожен з яких виконує унікальні функції, необхідні для ефективного аналізу зображень [20-22]. Ці шари взаємодіють для вилучення ознак, їх обробки і, зрештою, класифікації. Основні типи шарів у CNN:

– згортковий шар. Цей шар є основою нейронної мережі. У цьому шарі використовуються невеликі фільтри для сканування зображення і виявлення різних ознак, таких як краї, кути або прості геометричні фігури. Кожен фільтр згортається по зображенню, виконуючи операцію згортки, щоб створити карти ознак, які містять інформацію про специфічні патерни в зображенні. Це дозволяє мережі «бачити» локальні структури на зображенні і поступово виділяти більш складні ознаки на глибших рівнях мережі;

– пулінговий шар використовується для зменшення розмірності карт ознак, що отримані після згорткових шарів. Це зменшує кількість параметрів і обчислень у мережі, зберігаючи найбільш важливі ознаки;

– повнозв’язні шари розташовані в кінці архітектури нейронної мережі і відповідають за остаточну класифікацію. У повнозв’язних шарах кожен нейрон пов’язаний з усіма нейронами попереднього шару, що дозволяє мережі об’єднати всю вилучену інформацію з попередніх згорткових і пулінгових шарів. Основна функція цього шару – узагальнення виділених ознак і їх перетворення в ймовірності для кожного можливого класу. На виході останнього повнозв’язного шару зазвичай використовується функція Softmax, яка перетворює значення у ймовірності класів, забезпечуючи вибір найбільш ймовірного класу для зображення.

Класична архітектура CNN (рис. 2.3) складається з компонентів, що працюють разом для ефективного аналізу зображень. Згорткові шари виділяють ознаки, пулінгові шари зменшують розмірність даних, а повнозв’язні шари виконують остаточну класифікацію. Разом ці компоненти створюють потужну систему для вирішення завдань комп’ютерного зору.

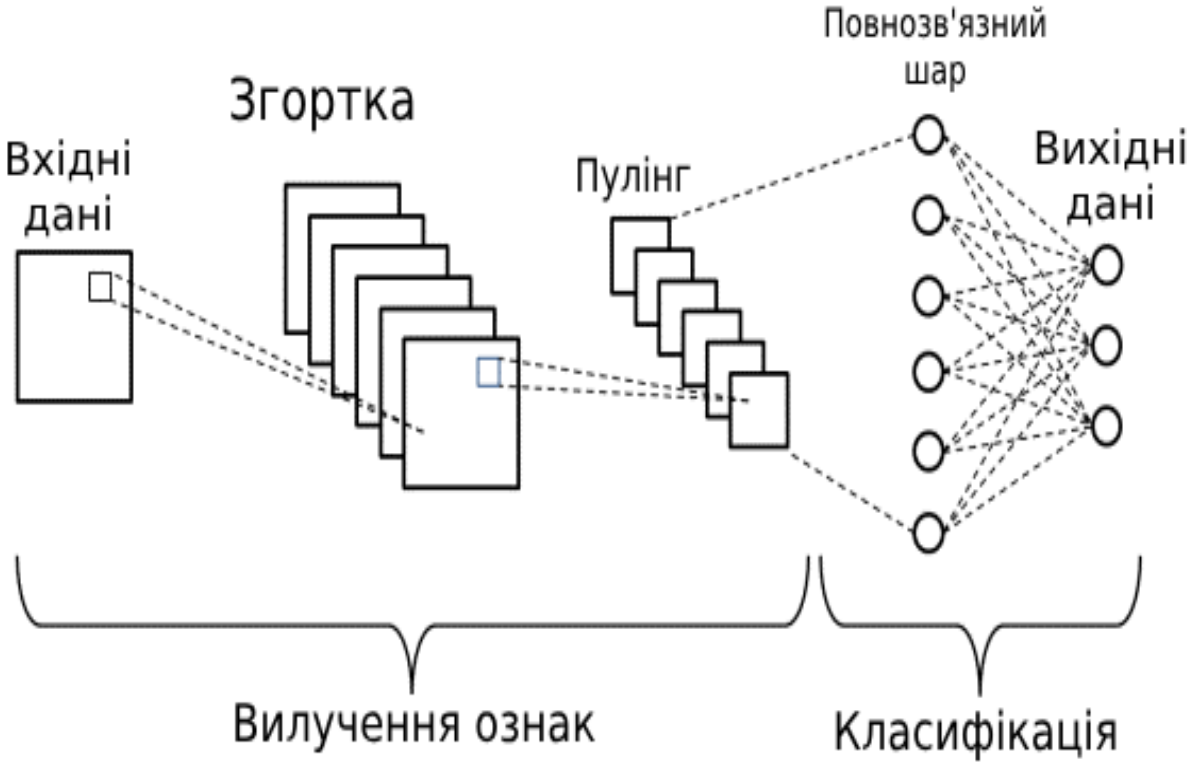

Рисунок 2.3 – Класична архітектура CNN

Процедура класифікації за допомогою CNN здійснюється за наступною схемою:

Крок 1. На вхід мережі подається зображення, яке представляється як тривимірний тензор (ширина, висота, кількість каналів). Наприклад, для кольорового зображення розміром 32×32 пікселів, вхід буде 32×32×3 (3 канали для кольорів RGB).

Крок 2. До вхідного зображення застосовується згортковий шар. У цьому шарі використовуються кілька фільтрів (наприклад, 5×5), які сканують зображення і виділяють локальні ознаки, такі як краї або текстури.

Крок 3. Кожен фільтр створює карту ознак, яка показує, де на зображенні присутні виділені фільтром ознаки. Зазвичай кілька фільтрів використовуються одночасно, що дозволяє отримати кілька карт ознак.

Крок 4. Застосовується пулінговий шар, зазвичай максимальний пулінг. Він зменшує розмір карт ознак, зберігаючи найважливішу інформацію. Наприклад, максимальний пулінг з вікном 2×2 вибирає максимальне значення

в кожній області розміру 2×2 і створює зменшену карту ознак. Це зменшення розмірності допомагає знизити кількість параметрів і обчислень, підвищуючи ефективність мережі.

Крок 5. Згорткові та пулінгові шари зазвичай повторюються кілька разів для виділення більш складних і абстрактних ознак на різних рівнях. На кожному етапі глибина мережі збільшується, і мережа навчається розпізнавати складніші патерни.

Крок 6. Після кількох рівнів згорткових і пулінгових шарів, тривимірні карти ознак перетворюються у вектор (одномірний масив) за допомогою процесу, званого флатенінг. Це потрібно для підготовки даних до повнозв’язних шарів.

Крок 7. Вектор передається на вхід повнозв’язного шару, де кожен нейрон з’єднаний з усіма нейронами попереднього шару. Це дозволяє об’єднати інформацію з усіх ознак, виділених у попередніх шарах.

Крок 8. Повнозв’язні шари працюють на рівні класифікації і допомагають мережі передбачати належність зображення до одного з класів.

Крок 9. Останній шар мережі – це вихідний шар, який використовує функцію Softmax для багатокласової класифікації. Ця функція перетворює значення на виході повнозв’язного шару в ймовірності для кожного класу.

Крок 10. На основі ймовірностей мережа робить остаточне передбачення, визначаючи клас, до якого належить зображення.

Крок 11. Під час навчання мережа проходить через цикли прямого і зворотного поширення помилки. У прямому проході мережа обчислює передбачення, а в зворотному – коригує свої ваги, щоб зменшити різницю між передбаченими значеннями і реальними мітками зображень.

Крок 12. Після навчання мережа оцінюється на тестовій вибірці, щоб перевірити її здатність класифікувати нові, невідомі раніше зображення. Якщо точність передбачень недостатня, архітектуру мережі можна вдосконалити, додавши або змінюючи кількість згорткових шарів, використовуючи регуляризацію або інші методи.

2.4 Vision Transformer

Глибоке навчання стало ключовим підходом у класифікації зображень завдяки здатності CNN автоматично вивчати високорівневі ознаки зображень. Із розвитком обчислювальних потужностей та збільшенням кількості даних дослідники почали створювати більш складні архітектури CNN, які дозволяють покращити точність моделей та їх ефективність.

ViT є відносно новою архітектурою (рис. 2.4) для задач комп'ютерного зору, яка використовує механізми трансформерів замість згорток [23-27]. ViT розбиває зображення на патчі, перетворюючи їх на послідовність, схожу на обробку тексту в моделях обробки природної мови. Хоча ViT вимагає великих наборів даних для тренування, вона демонструє високу продуктивність, перевершуючи традиційні CNN на деяких задачах.

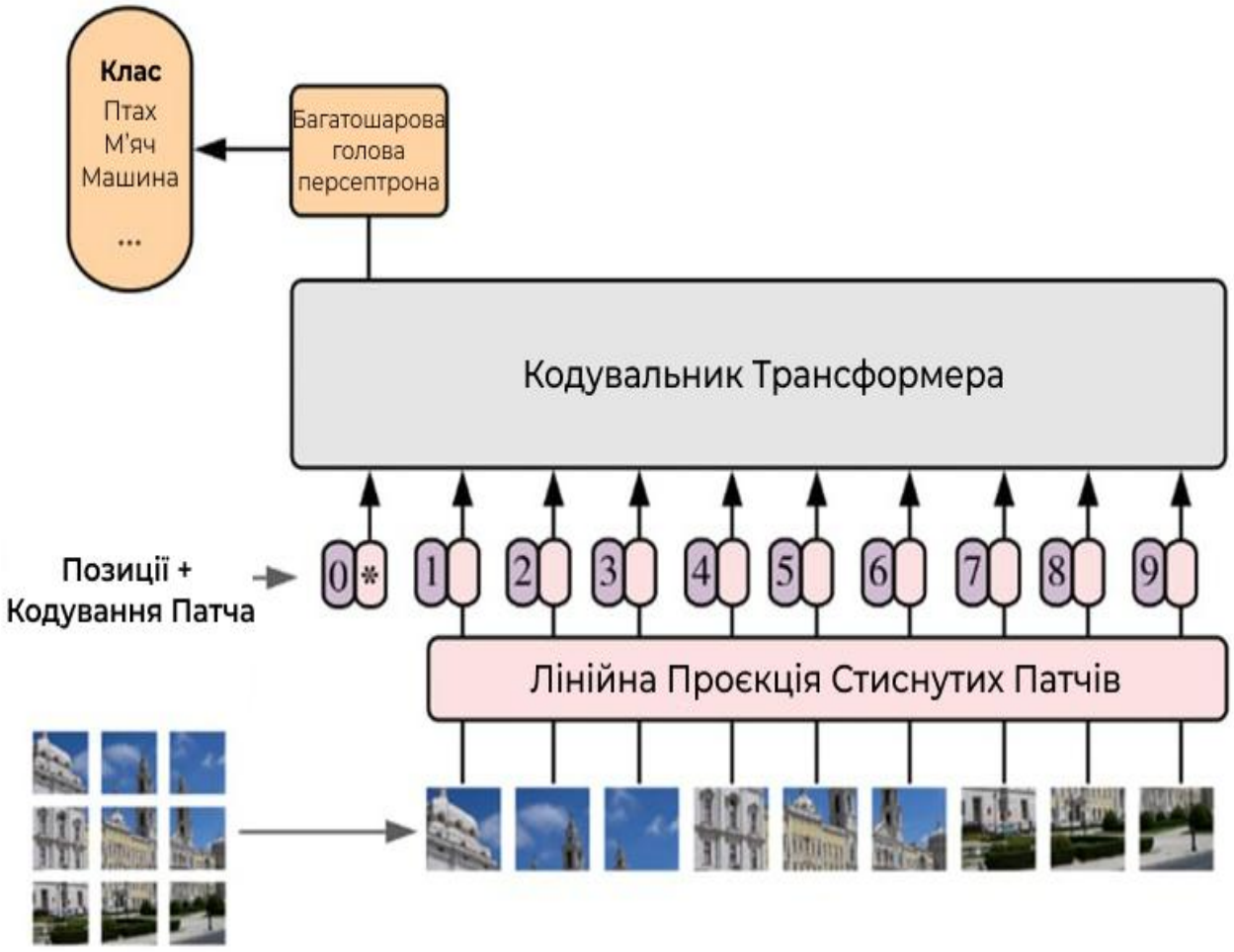

Рисунок 2.4 – Архітектура Vision Transformer

Загальна структура архітектури ViT складається з наступних кроків:

Крок 1. Розділення зображення на фрагменти фіксованих розмірів.

Крок 2. Стиснення фрагментів зображення.

Крок 3. Створення низьковимірних лінійних вставок зі стиснутих фрагментів зображення.

Крок 4. Додавання позиційних векторів.

Крок 5. Подача послідовності як вхідних даних до трансформера SOTA.

Крок 6. Попередньо навчіть модель ViT з мітками зображень, а потім повністю під контролем на великому наборі даних.

Крок 7. Налаштування вихідного набору даних для класифікації зображень.

Механізми трансформерів (рис. 2.5) замість роботи з окремими пікселями, ViT розбиває зображення на невеликі патчі. Кожен патч обробляється як окремий елемент, подібно до токенів у тексті, після чого перетворюється у вектор ознак. Це дозволяє моделі працювати з послідовністю патчів, що є більш зручним для трансформерів.

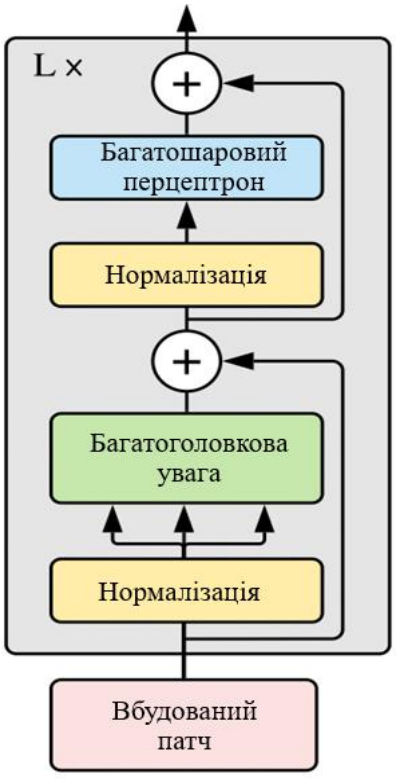

Рисунок 2.5 – Схема механізму трансформера

Шар самоуваги розраховує ваги уваги для кожного пікселя зображення на основі його взаємозв’язку з усіма іншими пікселями, тоді як шар подачі вперед застосовує нелінійну трансформацію до виходу шару самоуваги. Механізм багатоголової уваги дозволяє моделі одночасно звертати увагу на різні частини вхідної послідовності.

ViT також містить додатковий шар векторизації патчів, який ділить зображення на патчі фіксованого розміру та перетворює кожен патч на вектор високої розмірності. Ці векторизовані патчі потім подаються до блоків трансформера для подальшої обробки.

Зображення розміром 224×224 пікселів потребує $2{,}5 \cdot 10^9$ порівнянь. Якщо ж розбити зображення розміром 224×224 пікселів на 256 фрагментів по 14×14 пікселів (рис. 2.6), то в такому випадку один шар уваги потребує більш керованого $256 \cdot 14^4$ $(9{,}8 \cdot 10^6)$ порівняння. Тобто, це зменшить кількість обчислень та зробить процес обробки зображень більш ефективним і дозволить використовувати модель з меншою витратою ресурсів. Це особливо важливо для великих зображень, які потребують значних обчислювальних потужностей.

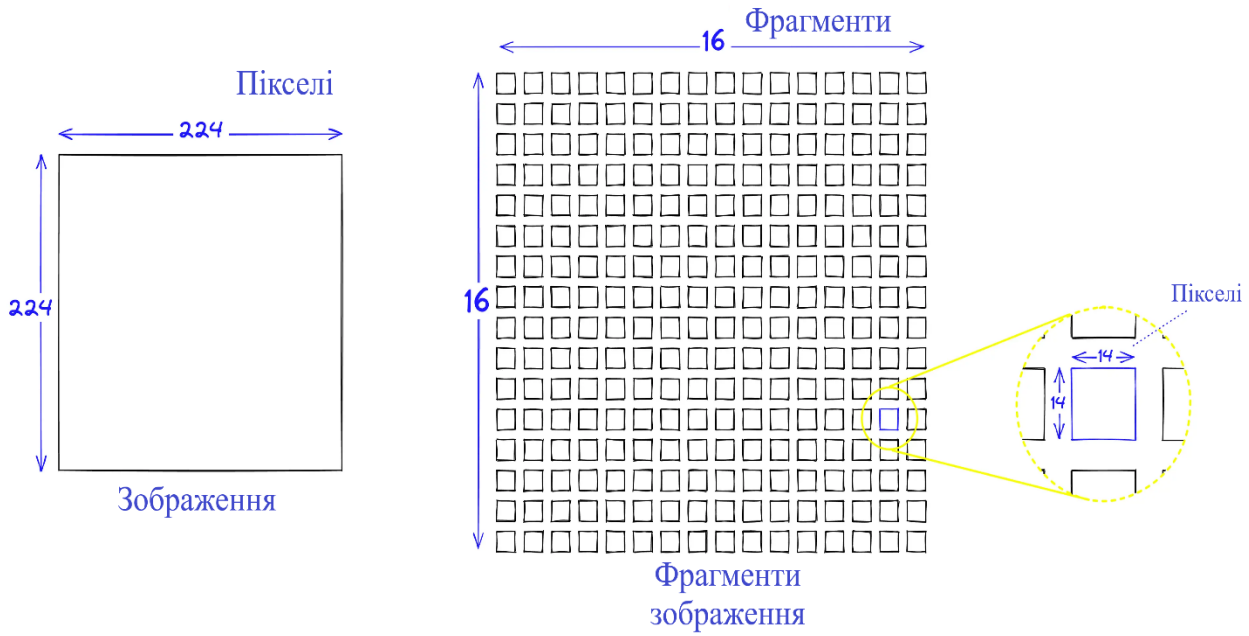

Рисунок 2.6 – Перетворення зображення розміром 224×224 пікселів у 256 фрагментів зображення розміром 14×14 пікселів

Остаточний вихід архітектури ViT – це передбачення класу (рис. 2.7), яке отримується шляхом пропускання виходу останнього блоку трансформера через класифікаційний шар. Зазвичай це складається з одного повнозв'язного шару.

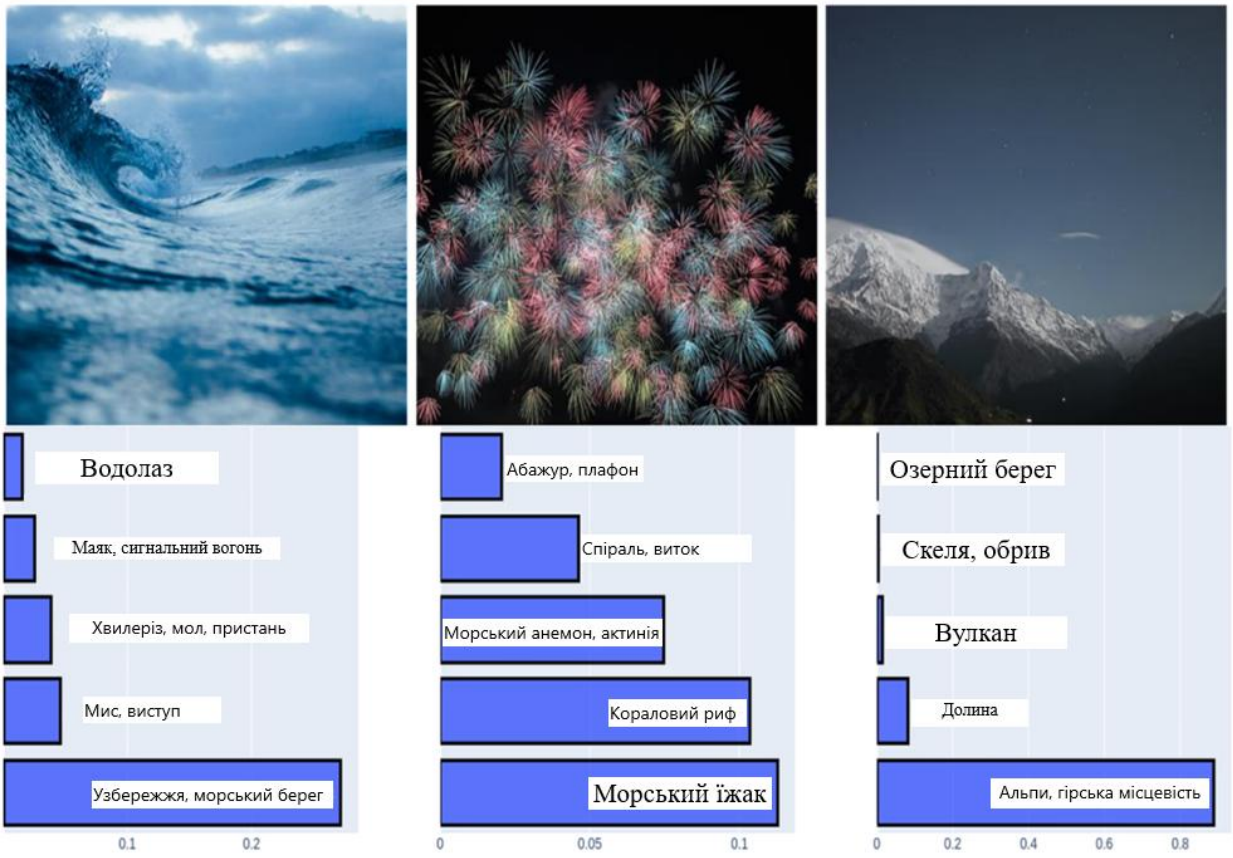

Рисунок 2.7 – Результати передбачень класів за допомогою ViT

Продуктивність моделі ViT залежить від таких рішень, як вибір оптимізатора, глибина мережі та гіперпараметри, специфічні для набору даних. У порівнянні з ViT, CNN легше оптимізувати.

Різниця в чистому трансформері полягає в поєднанні трансформера з фронтальним шаром CNN. Зазвичай на початку ViT використовують згортку 16×16 з кроком 16. Натомість, згортка 3×3 з кроком 2 підвищує стабільність та точність.

CNN перетворює базові пікселі на карти ознак. Далі токенізатор перетворює карту ознак на послідовність токенів, які подаються до

трансформера. Трансформер потім застосовує механізм уваги для створення послідовності вихідних токенів.

На малих і середніх наборах даних ViT не показують порівнянної продуктивності, але вони перевершують методи, що базуються на CNN на дуже великих наборах, що продемонстровано на рисунку 2.8. Це зумовлено тим, що CNN ефективніше кодують локальну інформацію завдяки використанню обмежених рецептивних полів для локальних областей зображення, на відміну від ViT.

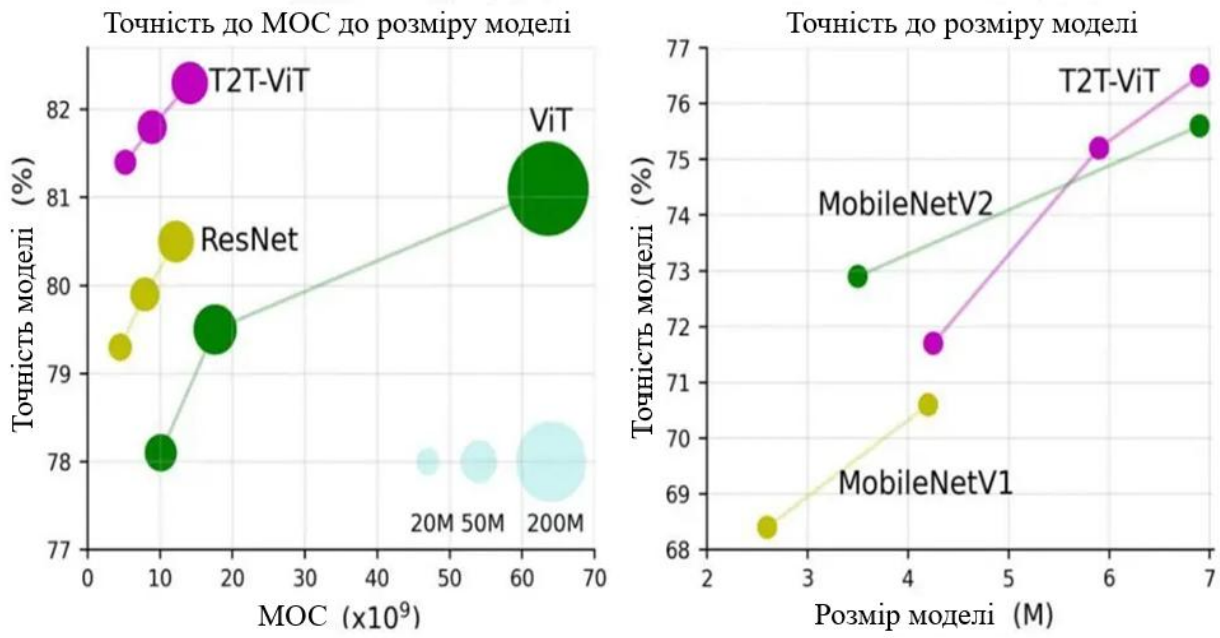

Рисунок 2.8 – Порівняння показників продуктивності ViT із ResNet і MobileNet під час навчання з нуля на ImageNet

У підсумку, проектор знову з’єднує вихідні токени з картою ознак. Це дозволяє аналізувати потенційно важливі деталі на рівні пікселів, що знижує кількість токенів, які потрібно вивчити, значно зменшуючи витрати.

Особливо якщо модель ViT навчена на великих наборах даних, що містять понад 14 мільйонів зображень, вона може перевершити CNN. Якщо ні, то найкращим вибором залишаються ResNet або EfficientNet. Модель Vision Transformer тренується на великому наборі даних ще до початку процесу тонкого налаштування. Для покращення тонкого налаштування на високих

роздільних здатностях виконується 2D представлення попередньо навчених позиційних векторів. Це необхідно, оскільки навчальні лінійні шари моделюють позиційні вектори.

Vision Transformer стикається з багатьма викликами, зокрема питаннями, пов’язаними з дизайном архітектури, узагальненням, стійкістю, інтерпретованістю та ефективністю.

Загалом, трансформери мають менше індуктивних упереджень порівняно з CNN і сильно залежать від великих обсягів даних для масштабного навчання. Тому якість даних значною мірою впливає на здатність трансформера до узагальнення та стійкості у завданнях комп’ютерного зору.

Хоча ViT демонструє відмінні результати у задачах класифікації зображень, наприклад, на наборах VTAB і CIFAR, його використання безпосередньо для виявлення об’єктів не перевершило результати CNN.

Крім того, повне розуміння причин, через які трансформери добре працюють на візуальних завданнях, залишається складним. Розробка ефективних моделей трансформерів для комп’ютерного зору, які можна використовувати на пристроях з обмеженими ресурсами, також є значним викликом.

Нейромережа здатна ефективно масштабуватися з ростом кількості шарів і параметрів, що дозволяє створювати надглибокі моделі для складних задач, в тому числі і для класифікації зображень. ViT не потребує попередньо визначеної структури згорток, що робить його гнучкішим при використанні різноманітних типів вхідних даних.

Використання трансформаторів зору для класифікації зображень має ряд недоліків. Навчання трансформаторів бачення є обчислювально дорогим. Будь-яке завдання, пов’язане з зображеннями, завжди дороге через великі розміри пікселів зображень. Це відбувається через те, що вони мають велику кількість параметрів. ViTs не такі ефективні, як CNN в обробці зображень. Це тому, що їм потрібно приділяти увагу кожній частині зображенні, навіть якщо це не є важливим для поставленого завдання.

Трансформатори зору – це перспективний новий підхід до класифікації зображень. Вони мають низку потенційних переваг, але також мають деякі недоліки. ViT продовжують розвиватися і, можливо, колись вони порівняються з CNN у різноманітних завданнях класифікації зображень.

2.5 Метрики вимірювання точності моделі

Точність – один із основних показників якості роботи моделі, проте в різних задачах можуть використовуватись й інші метрики. Оцінка точності – це процес вимірювання, наскільки добре модель виконує свою задачу на наборі тестових даних [28, 29].

Для початку необхідно розподілити дані на тренувальну, валідаційну та тестову вибірки. Тренувальна вибірка використовується для навчання моделі, допомагаючи їй знайти оптимальні ваги і параметри. Валідаційна вибірка, в свою чергу, слугує для налаштування гіперпараметрів моделі і моніторингу її продуктивності під час навчання. Тестова вибірка є набором даних, який модель ніколи не бачила під час навчання і ця вибірка використовується для оцінки здатності моделі узагальнювати результати на нові дані.

Як було зазначено раніше, точність – це найпростіша і найпоширеніша метрика для задач класифікації. Вона визначає відсоток правильно передбачених класів з усіх передбачень.

Простим прикладом для пояснення розрахунку точності моделі є класифікація даних на два класи: «Positive» та «Negative». Для оцінки її результатів використовується матриця плутанини (табл. 2.1), яка відображає чотири можливі варіанти передбачень: True Positive (TP) – випадки правильної класифікації позитивного класу; True Negative (TN) – випадки правильної класифікації негативного класу; False Positive (FP) – негативний клас помилково класифікується як позитивний; False Negative (FN) – коли позитивний клас класифікується як негативний.

Таблиця 2.1 – Загальний вид матриці плутанини

	Прогноз: Positive	**Прогноз: Negative**
Фактичне: Positive	True Positive (TP)	False Negative (FN)
Фактичне: Negative	False Positive (FP)	True Negative (TN)

Точність обчислюється за наступною формулою:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (2.1)$$

де $Accuracy$ – точність.

Припустимо, що модель класифікує 100 зразків. Із них: 50 прикладів – позитивні зразки і ще 50 прикладів це негативні зразки.

Модель правильно класифікувала 40 позитивних як позитивні (TP) і 45 негативних як негативні (TN). Модель помилилася у 5 негативних випадках коли зразки класифіковано як позитивні (FP) і у 10 позитивних класифікованих як негативні (FN). Точність в даному випадку буде розраховуватися наступним чином:

$$Accuracy = \frac{40 + 45}{40 + 45 + 5 + 10} = \frac{85}{100} = 0{,}85. \quad (2.2)$$

Точність позитивних передбачень знаходять як частку правильних позитивних передбачень серед усіх передбачених позитивних випадків

$$Precision = \frac{TP}{TP + FP}, \quad (2.3)$$

де $Precision$ – точність позитивних передбачень.

Чутливість знаходять як частку правильно передбачених позитивних випадків серед усіх справжніх позитивних випадків.

$$Recall = \frac{TP}{TP + FN}, \tag{2.4}$$

де $Recall$ – чутливість.

F1-Score використовується в задачах класифікації, коли є необхідність збалансувати точність позитивних передбачень і чутливість, особливо у випадках, коли дані мають дисбаланс класів або коли помилкові позитивні та негативні передбачення мають різні наслідки. F1-Score є гармонійним середнім між точністю та повнотою і показує загальну продуктивність моделі і виглядає наступним чином:

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}, \tag{2.5}$$

де F_1 – гармонічне середнє між Precision і Recall.

F1-Score також може використовуватись для оцінки моделі, коли потрібно запобігти перенавчанню. Якщо модель має високу точність на тренувальних даних, але низьку повноту на тестових даних, це свідчить про перенавчання. У таких випадках F1-Score допомагає оцінити загальну ефективність моделі.

Втрати – це метрика, яка вимірює, наскільки добре модель виконує свою задачу. Вона показує різницю між передбаченими і фактичними значеннями, які ми хочемо передбачити. Втрати використовуються для оцінки продуктивності моделі під час навчання. Мета навчання полягає в мінімізації функції втрат, що означає, що ми прагнемо досягти кращих передбачень. Якщо втрата зменшується, це свідчить про покращення продуктивності моделі. Якщо вона перестає зменшуватися або починає зростати, це може бути знаком переобладнання або недостатнього навчання.

На початку навчання втрата зазвичай висока, а точність дуже мала оскільки модель ще не навчена і її передбачення далекі від правильних. Зі збільшенням кількості епох втрата знижується, а точність зростає, оскільки модель адаптується до даних і покращує свої передбачення. Після певного числа епох метрики продуктивності моделі перестають суттєво змінюватися і це вказує на те, що модель досягла свого максимуму продуктивності на даному наборі даних. На рисунку 2.9 показано залежність втрат і точності від кількості епох на тренувальному і валідаційному наборах даних.

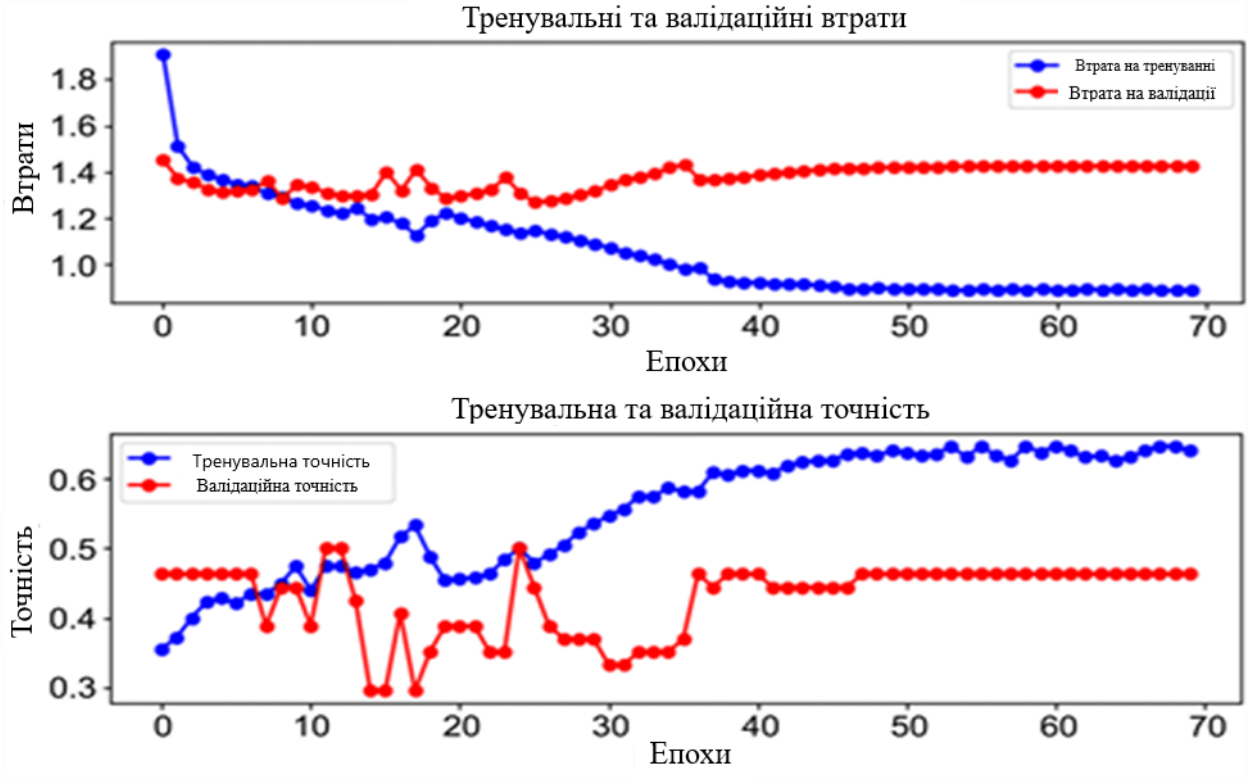

Рисунок 2.9 – Залежність втрат і точності від кількості епох

2.6 Градієнтний вибух

Градієнтний вибух – це проблема, яка може виникати під час навчання глибоких нейронних мереж [30]. Вона характеризується надмірним збільшенням значень градієнтів, що може призвести до нестабільності процесу навчання, а також до чисельних помилок.

Причини вибухів можуть бути досить різними. Наприклад, коли нейронна мережа має багато шарів, градієнти, які передаються назад через мережу, можуть накопичуватися, що призводить до їх різкого збільшення або ініціалізація ваг з великими значеннями може викликати зростання градієнтів.

Зазвичай, вказують на градієнтні вибухи дуже великі або нескінченні значення під час навчання або різке збільшення значення функції втрат. Градієнтний вибух ускладнює навчання моделі, оскільки ваги можуть змінюватися занадто швидко, що призводить до нестабільних результатів і поганої узагальнюючої здатності.

Існує кілька способів запобігання градієнтного вибуху:

– правильна ініціалізація ваг моделі;

– зменшення глибини мережі;

– застосування нормалізації пакетів;

– використання технік зниження ваг;

– використання адаптивних алгоритмів оптимізації;

– регуляризація.

Використання цих методів може суттєво поліпшити процес навчання нейронних мереж, запобігаючи градієнтному вибуху та забезпечуючи стабільні результати. Застосування цих підходів у комплексі може значно підвищити успішність навчання і зробити моделі більш надійними для вирішення складних завдань у різних сферах.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ НЕЙРОМЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи було розроблено метод для класифікації зображень на основі нейронних мереж із застосуванням сучасних методів обробки даних. Для реалізації було обрано середовище PyCharm 2024.2.1, оскільки воно є одним із найпопулярніших і потужних IDE для програмування на мові Python [31]. PyCharm забезпечує зручну і продуктивну роботу з Python завдяки своїм численним інструментам, таким як автодоповнення коду, інтегроване відлагодження, підтримка віртуальних середовищ і керування пакетами. Інтерфейс застосунку PyCharm показано на рисунку 3.1.

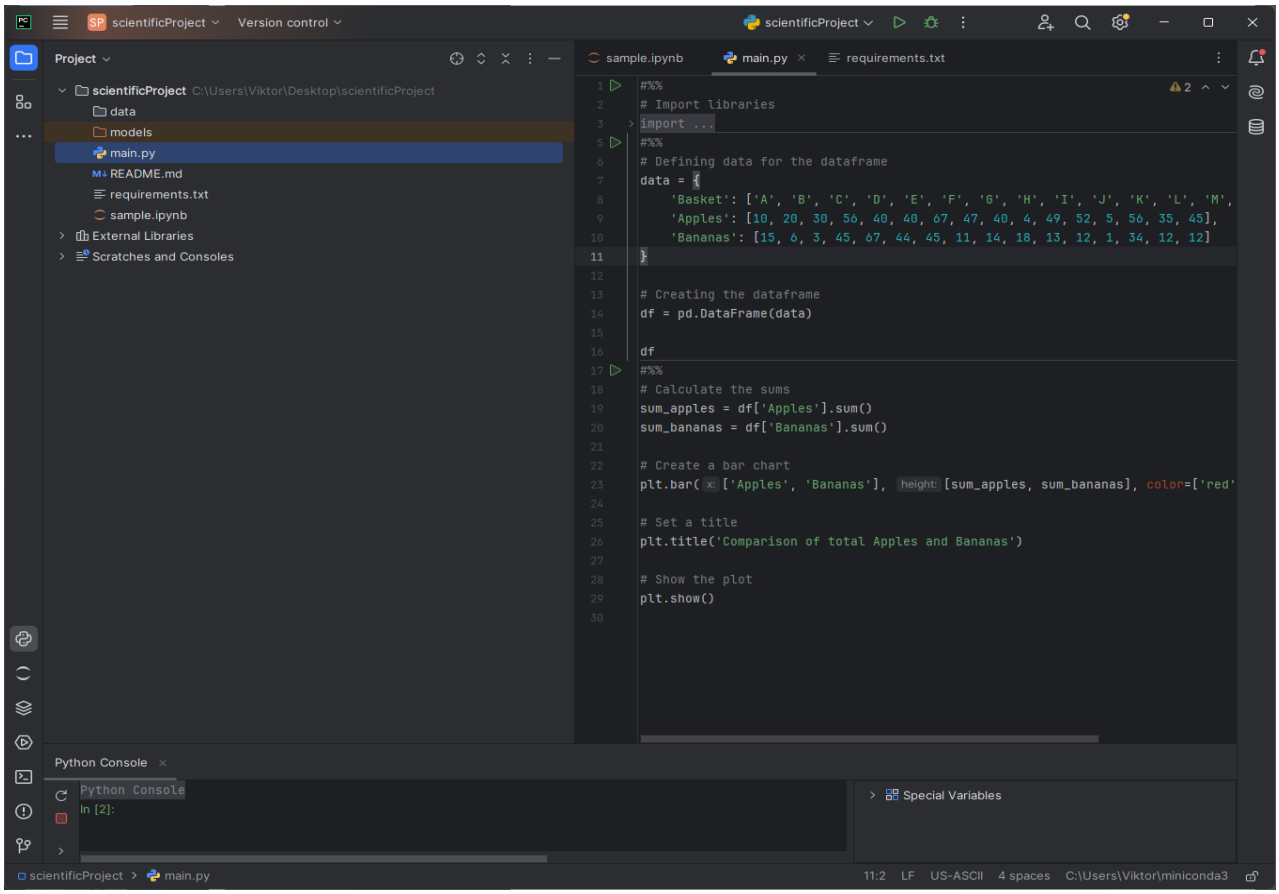

Рисунок 3.1 – Приклад інтерфейсу PyCharm

Це середовище розробки також пропонує функції для роботи з базами даних, фреймворками та науковими бібліотеками, що робить його ідеальним вибором для створення різноманітних застосунків на Python, включаючи десктопні, вебзастосунки та скрипти для автоматизації процесів. Також у середовище розробки PyCharm може бути інтегрований Jupyter Notebook, що дозволяє використовувати всі можливості Jupyter, такі як інтерактивність та візуалізація, у знайомому для розробників середовищі.

Як було зазначено раніше, для створення застосунку було обрано мову Python [32]. Це можна обґрунтувати тим, що Python є потужною і сучасною мовою програмування, яка добре підходить для розробки застосунків, зокрема для класифікації зображень. Вона має простий і зрозумілий синтаксис, що полегшує розробку, а також підтримку об’єктно-орієнтованого програмування. Використання Python дозволить зручно та ефективно реалізувати функціонал класифікації зображень у застосунку, забезпечуючи високу точність і продуктивність моделей.

Python став надзвичайно популярним серед фахівців із машинного навчання з кількох причин. Як відомо, його синтаксис часто описують як «елегантний» і «математичний», що полегшує реалізацію математичних ідей. Це дає можливість використовувати Python для складних математичних завдань без необхідності довго вивчати нову мову програмування. Python також відомий своєю простотою, що робить його зручним для різноманітних прикладних сфер, зокрема для машинного навчання. Багато програмістів називають Python оптимальним «компромісом між складністю та продуктивністю», а його синтаксис інтуїтивно зрозумілий у порівнянні з деякими іншими мовами.

Серед переваг Python – широкий набір спеціалізованих інструментів для машинного навчання, включно з такими бібліотеками, як NumPy, які спрощують реалізацію задач. Суттєву допомогу надає й бібліотека scikit-learn, яка охоплює 72 модулі для роботи з машинним навчанням на Python. Порівняно з мовами Java, Ruby on Rails, C або Perl, Python вважають більш

зручним для застосування в машинному навчанні. Дехто критикує його як «іграшкову мову», проте численні користувачі визнають його цілком функціональною альтернативою завдяки доступному синтаксису та універсальності.

Python сприяє спільній розробці й реалізації проектів, а як мова загального призначення може виконувати різні функції, що важливо для комплексних задач у машинному навчанні. Крім того, завдяки великій спільноті підтримки, Python стає ще більш привабливим вибором для спеціалістів у сфері технологій.

Для роботи з класифікацією зображень на Python існує велика кількість потужних бібліотек, що робить його ідеальним вибором для реалізації алгоритмів класифікації зображень. Наприклад, бібліотека TensorFlow надає широкий спектр функціональності для побудови, навчання та оцінки нейронних мереж, що використовуються для класифікації зображень. Інші бібліотеки, такі як Keras і PyTorch, також пропонують інструменти для глибокого навчання та аналізу зображень, включаючи підтримку обробки великих наборів даних, налаштування різних моделей, зокрема, ResNet чи EfficientNet, та роботу з попередньо навченими моделями. Використання середовища PyCharm дозволить легко інтегрувати ці бібліотеки в проєкт і ефективно використовувати їх функціонал.

Операційна система Windows надає розробникам додаткові інструменти для роботи з Python, зокрема через бібліотеки, які підтримують апаратне прискорення, наприклад, CUDA для NVIDIA GPU, що є критично важливим для виконання обчислювальних задач, зокрема в галузі машинного навчання, обробки зображень та інших ресурсномістких обчислень. У мові розробки Python доступні прості та ефективні засоби для роботи з класифікацією зображень, які дозволяють швидко створювати, навчати та використовувати моделі машинного навчання.

3.1.1 Jupyter Notebook

Jupyter Notebook – це інтерактивне середовище для аналізу даних та виконання програмного коду, яке особливо популярне серед науковців, аналітиків даних та розробників машинного навчання (рис. 3.2). Воно дозволяє виконувати код поетапно та отримувати миттєвий зворотний зв'язок, що робить його ідеальним для експериментів з даними, візуалізацій і моделювання [33]. Взаємодія між Python і Jupyter Notebook побудована на основі так званих «ядер», що виконують код Python і повертають результати безпосередньо у середовище.

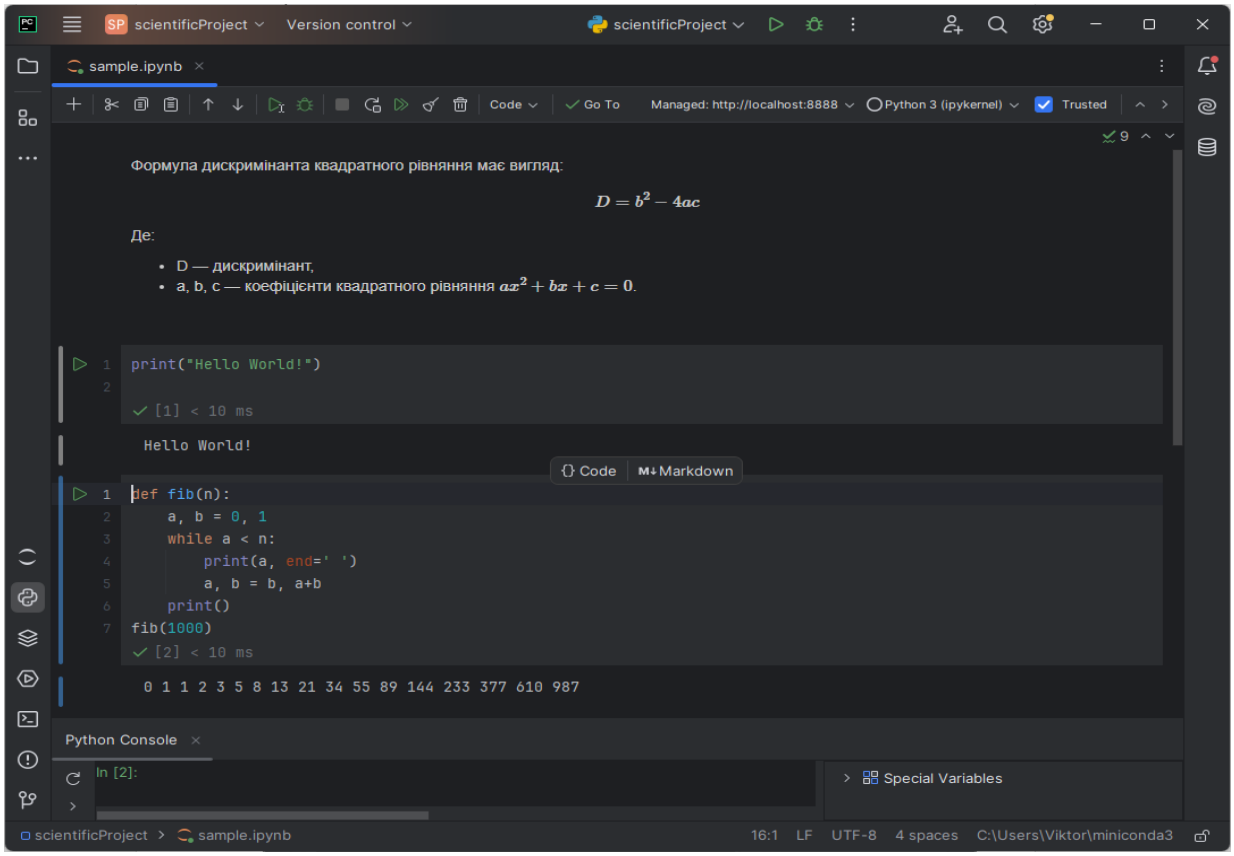

Рисунок 3.2 – Інтерфейс застосунку у Jupyter Notebook

Jupyter Notebook має простий і інтуїтивно зрозумілий інтерфейс, який складається з кількох ключових компонентів, але основний вміст Jupyter Notebook організовано в комірках. Комірки поділяють на два види: комірки для коду та комірки для розмітки.

В комірках для коду можна писати та виконувати код на Python. Коли запускається комірка, результат її виконання з'являється безпосередньо під коміркою. Оскільки можна запускати окремі комірки коду, а не весь код одразу, то це дає змогу легко експериментувати з кодом і вносити зміни в реальному часі. Комірку для написання коду показано на рисунку 3.3.

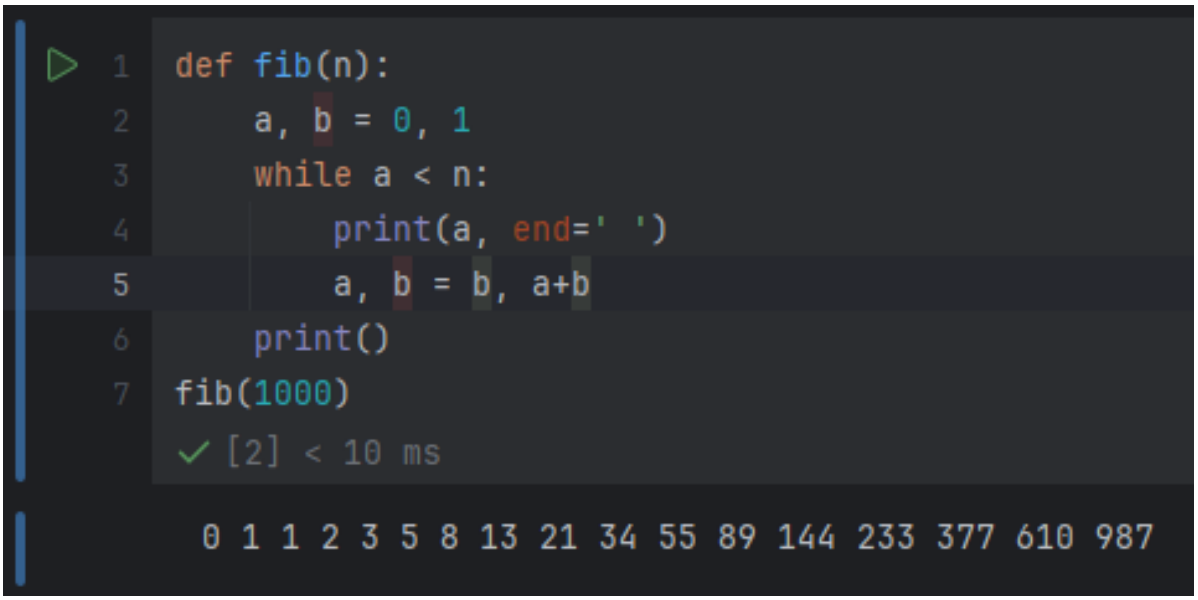

Рисунок 3.3 – Комірка для написання коду

В комірках для розмітки зазвичай описують документацію коду. У таких комірках можна використовувати форматування, заголовки, списки, зображення та навіть LaTeX для математичних рівнянь. Комірку для розмітки показано на рисунку 3.4.

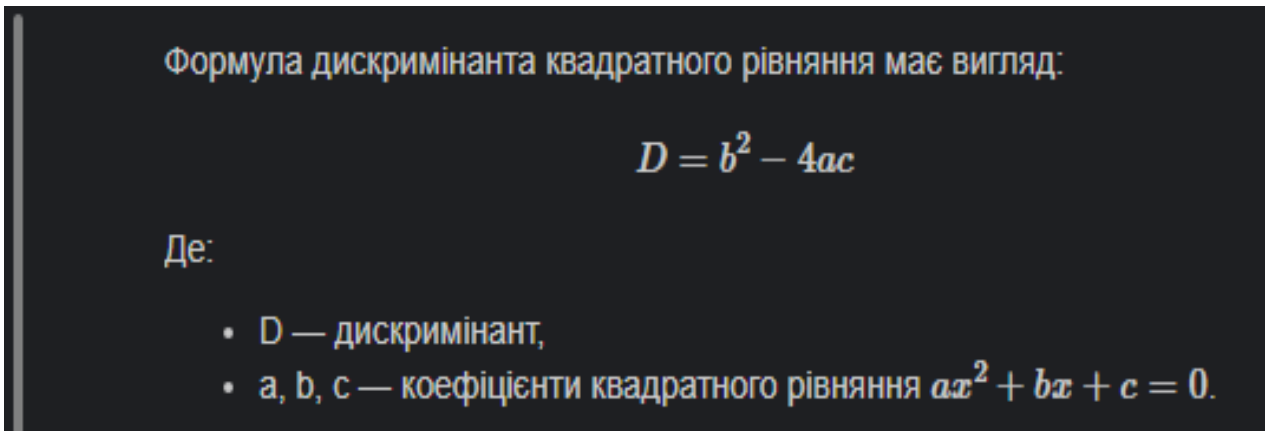

Рисунок 3.4 – Комірка для розмітки

3.1.2 Discord-бот

Discord – це безкоштовна платформа для спілкування, яка поєднує в собі функції текстових чатів, голосових дзвінків та відеоконференцій (рис. 3.5). Спочатку розроблений для геймерів, Discord швидко набув популярності серед різноманітних спільнот, включаючи професіоналів, освітні групи, творчі колективи та багато інших.

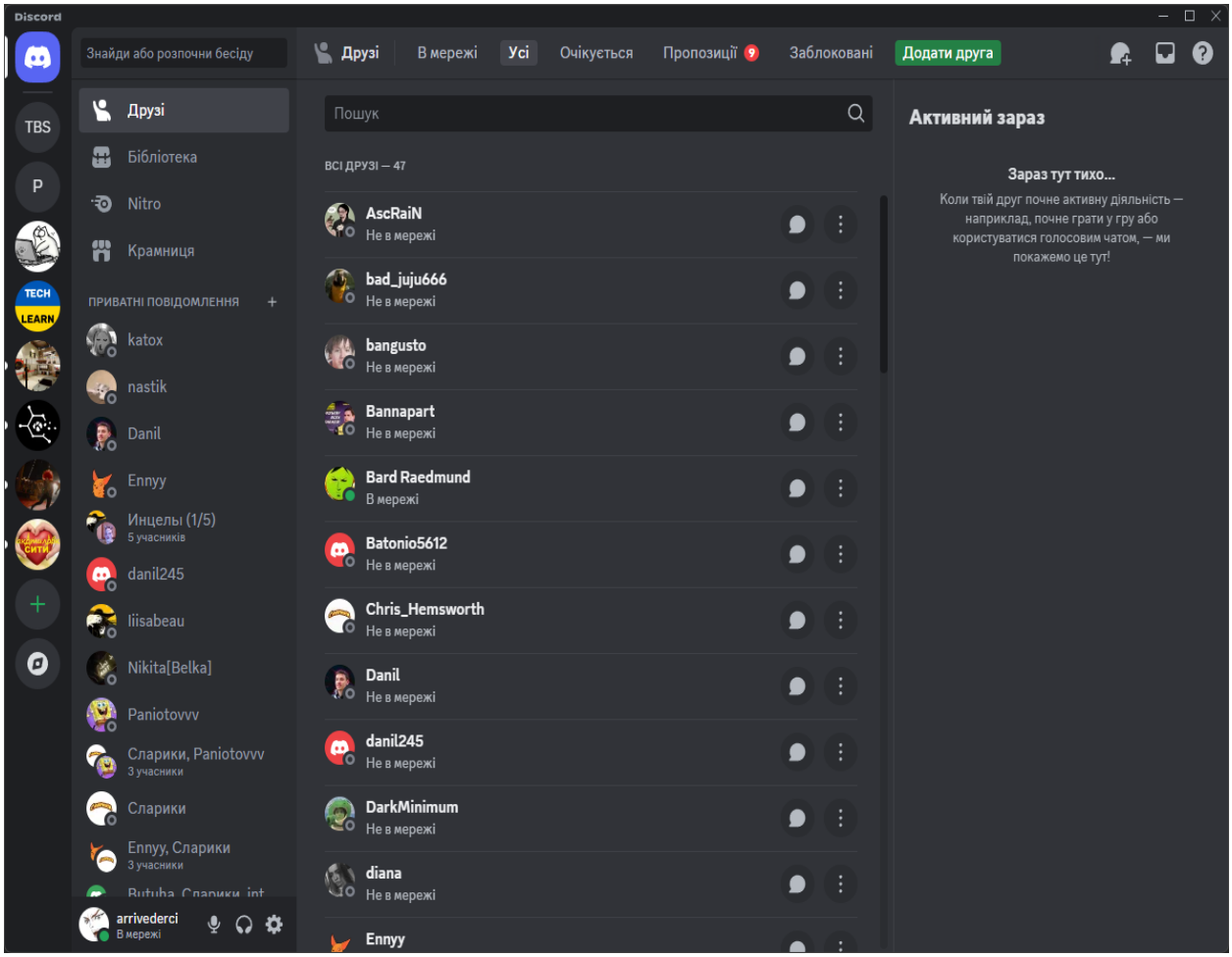

Рисунок 3.5 – Зовнішній вигляд застосунку Discord

Завантаження застосунку у вигляді Discord-бота [34] чудове рішення багатьох проблем, пов’язаних із доступністю, взаємодією та масштабованістю сервісів, особливо у сфері машинного навчання та обробки зображень.

Discord є однією з найпопулярніших платформ для спілкування серед геймерів, професіоналів та спільнот різного спрямування. Це означає, що бот

автоматично стає доступним для великої аудиторії, яка вже активно використовує цю платформу. Таким чином, інтегруючи застосунок у Discord, забезпечується легкий доступ користувачів до функціональності бота без необхідності встановлювати додаткові програми або переходити на інші платформи. Платформа надає зручні інструменти для інтерактивної взаємодії. Користувачі можуть легко надсилати зображення, текстові запити або використовувати команди для взаємодії з ботом. У випадку з ботом, що використовує Vision Transformer (ViT) та набір даних CIFAR-10, користувачі можуть просто завантажити зображення у чат, а бот миттєво обробить його та поверне результат класифікації. Це забезпечує швидкий та зручний спосіб отримання результатів без зайвих кроків.

Discord надає потужний API та бібліотеки, які спрощують розробку та розгортання ботів. Бібліотеки, такі як Discord.py, дозволяють легко інтегрувати моделі машинного навчання з Discord, обробляти події, надсилати повідомлення та керувати взаємодією з користувачами. Це значно спрощує процес розробки, дозволяючи зосередитися на вдосконаленні моделі та її функціональності, а не на складних аспектах інтеграції з платформою. Discord підтримує масштабованість та надійність. Бот може обслуговувати сотні або навіть тисячі користувачів одночасно без значних витрат на інфраструктуру. Це особливо важливо для застосунків, що потребують обробки великої кількості запитів або роботи з великими обсягами даних, наприклад, класифікація зображень у реальному часі.

Інтеграція з Discord дозволяє додавати додаткові функції та розширювати можливості бота, тобто є змога створювати команди для різних типів запитів, додавати функції аутентифікації, налаштовувати ролі та права доступу, а також інтегруватися з іншими сервісами та API для розширення функціональності застосунку.

Крім того, Discord забезпечує високу залученість користувачів через функції сповіщень, повідомлень у реальному часі та інтерактивних елементів, таких як реакції на повідомлення чи інтерактивні меню. Це створює більш

інтерактивний та привабливий досвід для користувачів, стимулюючи їх до активної взаємодії з ботом. Discord є крос-платформеним сервісом, доступним на різних пристроях – від комп'ютерів до мобільних телефонів. Це означає, що користувачі можуть взаємодіяти з ботом будь-де та будь-коли, що підвищує зручність використання та доступність сервісу.

Discord надає можливості для аналізу та моніторингу активності бота через свої аналітичні інструменти та логи. Це дозволяє відстежувати використання бота, виявляти потенційні проблеми та оптимізувати його роботу для покращення продуктивності та користувацького досвіду.

3.2 Бібліотеки для реалізації методів

Вибір відповідних бібліотек для реалізації методів машинного навчання є критично важливим. Вибір бібліотек з хорошою документацією та простим API може значно полегшити процес розробки. Також вибір бібліотек, які спрощують розробку, дозволяє зосередитися на ключових аспектах проєкту, зменшуючи час на написання коду.

Основною бібліотекою для реалізації буде PyTorch [35], оскільки вона є популярною бібліотекою для глибокого навчання, що забезпечує гнучкість і простоту використання. Вона дозволяє створювати та тренувати нейронні мережі, пропонує автоматичне диференціювання і підтримує обчислення на GPU.

PyTorch забезпечує основні компоненти для створення, навчання і тестування нейронних мереж, а також для реалізації алгоритмів глибокого навчання. Бібліотека також пропонує динамічний підхід до побудови моделей, що дозволяє легко змінювати архітектуру в процесі виконання. Автоматичне диференціювання є однією з ключових функцій PyTorch, що спрощує реалізацію зворотного розповсюдження помилки.

PyTorch має вбудовану підтримку GPU, що дозволяє суттєво прискорити обчислення. CUDA – це платформа паралельних обчислень та API, розроблена NVIDIA, яка дозволяє програмістам використовувати GPU для загальних обчислень. PyTorch інтегрується з CUDA, що дозволяє розробникам безпосередньо використовувати GPU для виконання обчислень.

GPU містять тисячі малих ядер, що дозволяє їм виконувати багато обчислень одночасно, що є особливо корисним для обробки великих обсягів даних. Під час навчання обчислення градієнтів і оновлення ваг можуть відбуватися паралельно, що значно зменшує час навчання моделі.

Також є багато бібліотек, що доповнюють PyTorch, забезпечуючи специфічні інструменти та моделі для роботи з зображеннями, але вони не можуть функціонувати без PyTorch, наприклад, Torchvision або Timm.

Torchvision – це підмодуль PyTorch, який надає інструменти для обробки зображень і роботи з наборами даних [36]. Він включає в себе популярні набори даних, такі як CIFAR10, MNIST та інші. Також бібліотека надає набір стандартних трансформацій, таких як обертання, зміна розміру, нормалізація, що дозволяє готувати зображення для подачі в модель. До того ж, Torchvision містить попередньо натреновані моделі, які можна використовувати для переносу навчання або для оцінки без потреби в значних обчисленнях.

Бібліотека OS – це стандартна бібліотека Python для взаємодії з операційною системою. Вона надає інструменти для роботи з файловою системою. Можливість створювати, видаляти та управляти файлами і папками все це змоги отримання інформації про середовище, що надає можливість доступу до змінних середовища, що може бути корисно для конфігурації середовища виконання. Також є функції для зміни поточної директорії та отримання списку файлів у каталозі.

Tf-keras є високорівневим API для TensorFlow, яке спрощує створення та тренування нейронних мереж. Вона забезпечує зручний інтерфейс для визначення моделей, додавання шарів, налаштування функцій втрат та оптимізаторів, а також підтримує різні методи навчання та валідації. Tf-keras

дозволяє швидко прототипувати та експериментувати з різними архітектурами моделей, одночасно використовуючи потужність та гнучкість TensorFlow для створення високоефективних рішень у сфері глибокого навчання.

Datasets від Hugging Face надає потужний та зручний інтерфейс для завантаження, обробки та управління великими наборами даних [37]. Бібліотека підтримує численні популярні датасети для різних задач, таких як класифікація, генерація тексту, переклад тощо, забезпечуючи ефективні методи фільтрації, аугментації та розподілу даних на тренувальні, валідаційні та тестові набори. Це дозволяє швидко інтегрувати дані в робочий процес та зосередитися на розробці моделей без зайвих труднощів з управлінням даними.

Бібліотека Transformers (також розроблена компанією Hugging Face) є одним з найпопулярніших та найпотужніших інструментів у сфері обробки природної мови та машинного навчання загалом [38]. Бібліотека надає доступ до широкого спектру переднавчених моделей трансформерів, таких як BERT, GPT, T5, RoBERTa, DistilBERT, та багатьох інших, які стали революціонерами у підходах до вирішення різноманітних завдань, пов’язаних з аналізом та генерацією тексту. Завдяки своїй гнучкості, масштабованості та простоті використання, Transformers стала незамінним інструментом для дослідників, розробників та компаній, що працюють у галузі штучного інтелекту.

Однією з ключових особливостей бібліотеки є її здатність забезпечувати легкий доступ до переднавчених моделей, що дозволяє користувачам швидко впроваджувати та адаптувати ці моделі під свої конкретні задачі без необхідності починати навчання з нуля. Це значно зменшує час та обчислювальні ресурси, необхідні для розробки ефективних моделей машинного навчання. Крім того, бібліотека підтримує як PyTorch, так і TensorFlow, що робить її універсальною та сумісною з різними екосистемами машинного навчання.

Transformers пропонує різноманітні API для завантаження, налаштування та використання моделей. Користувачі можуть легко

завантажувати переднавчені моделі з Hugging Face Model Hub, який містить тисячі моделей, натренованих на різних датасетах та для різних задач. Це забезпечує широкий вибір моделей для класифікації тексту, генерації тексту, перекладу, суммаризації, відповіді на запитання, аналізу сентименту та багатьох інших застосувань.

Бібліотека також підтримує тонке налаштування моделей (fine-tuning), що дозволяє адаптувати їх до специфічних доменів або завдань. Це особливо корисно, коли необхідно покращити продуктивність моделі на специфічному наборі даних або коли завдання вимагає певної експертизи, яку модель може набувати під час додаткового навчання. transformers забезпечує зручний інтерфейс для налаштування параметрів навчання, таких як швидкість навчання, розмір батча, кількість епох та інші гіперпараметри, що дозволяє оптимізувати процес навчання під конкретні потреби користувача.

Однією з вагомих переваг Transformers є її інтеграція з іншими бібліотеками та інструментами Hugging Face, такими як Datasets для ефективного завантаження та обробки даних, Tokenizers для швидкого та гнучкого токенізації тексту та Accelerate для спрощення розподіленого навчання на різних обчислювальних ресурсах. Це створює екосистему, яка дозволяє безперешкодно переходити від завантаження даних до навчання, оцінки та розгортання моделей.

Transformers також активно підтримується спільнотою, що забезпечує регулярні оновлення, виправлення помилок та додавання нових функціональних можливостей. Документація бібліотеки є детальною та зрозумілою, містить численні приклади та посібники, які допомагають користувачам швидко освоїтися з її функціоналом. Hugging Face організовує форуми та спільноти, де користувачі можуть обмінюватися досвідом, ставити запитання та отримувати підтримку від інших членів спільноти та розробників.

Ще однією важливою характеристикою є можливість розгортання моделей у різних середовищах, включаючи локальні сервери, хмарні

платформи та мобільні пристрої. Це дозволяє використовувати потужні моделі трансформерів у реальних застосунках, забезпечуючи високу продуктивність та масштабованість. Наприклад, моделі можуть бути інтегровані в чат-ботів, системи рекомендацій, автоматизовані системи відповіді на запитання, інструменти для аналізу тексту в реальному часі та багато іншого.

Безпека та етичні аспекти також враховуються в Transformers. Hugging Face активно працює над виявленням та усуненням потенційних упереджень у моделей, а також над забезпеченням прозорості та відповідальності у використанні моделей машинного навчання. Це включає надання інструментів для аналізу та оцінки моделей з точки зору їхньої справедливості, зрозумілості та надійності.

Бібліотека Discord є досить потужним інструментом для розробки ботів та інтеграції з платформою Discord [39]. Вона служить обгорткою над Discord API, надаючи розробникам зручний інтерфейс для взаємодії з різноманітними функціями Discord, такими як відправка повідомлень, управління каналами, обробка подій та багато іншого. Завдяки цій бібліотеці можна легко створювати ботів, які реагують на різні дії користувачів, автоматизують задачі та забезпечують інтерактивний досвід спілкування. Бібліотека підтримує асинхронне програмування, що дозволяє ефективно обробляти численні запити одночасно, забезпечуючи високу продуктивність та швидкість відповіді бота. Вона також забезпечує доступ до різноманітних можливостей Discord, включаючи управління ролями, інтеграцію з вебхуками, обробку файлів та багато іншого, що робить її незамінним інструментом для розробників, які прагнуть створити функціональні та інтерактивні застосунки на базі Discord.

Модуль Discord.ext.commands є розширенням бібліотеки Discord і надає більш структурований підхід до створення ботів з використанням командної архітектури. Цей модуль дозволяє визначати команди за допомогою декораторів, що спрощує організацію та управління функціональністю бота. Завдяки Discord.ext.commands можна легко реалізовувати команди з

параметрами, підкоманди, групи команд та інші складні структури, що робить бот більш гнучким та масштабованим. Крім того, цей модуль підтримує автоматичну обробку помилок, валідацію вхідних даних та управління правами доступу, що сприяє створенню більш надійних та безпечних застосунків. Використовуючи Discord.ext.commands, можна зосередитися на реалізації бізнес-логіки бота, залишаючи обробку низькорівневих деталей бібліотеці, що значно спрощує процес розробки та підтримки. Таким чином, поєднання бібліотеки Discord та модуля Discord.ext.commands створює потужний інструментарій для створення функціональних, ефективних та інтерактивних застосунків на платформі Discord.

3.3 Програмна реалізація

Розробка застосунку, який спрощує дослідження нейромережі для класифікації зображень, включає кілька ключових етапів, кожен з яких відіграє важливу роль у створенні ефективного та надійного інструменту. Нижче наведено детальний покроковий опис процесу програмної реалізації цього застосунку:

Крок 1. Налаштування середовища розробки. Створення віртуального середовища за допомогою інструментів, таких як venv або conda, допомагає ізолювати залежності проекту та уникнути конфліктів між пакетами. Це дозволить встановити усі необхідні бібліотеки, включаючи Discord.py, Torch, Transformers, PIL, IO, Requests тощо, та забезпечить доступ до всіх інструментів, необхідних для розробки та запуску застосунка.

Крок 2. Підготовка та навчання моделі ViT на наборі даних CIFAR-10. Цей крок передбачає вибір відповідної архітектури моделі, її завантаження з використанням бібліотек машинного навчання, таких як Torch та Transformers, та донавчання на специфічному наборі даних для досягнення високої точності класифікації. Після завершення навчання та налаштування, модель досягла

високої точності класифікації на рівні 97,5% (рис. 3.6). Такий рівень точності демонструє здатність моделі ефективно розпізнавати та класифікувати різноманітні об’єкти, забезпечуючи надійну основу для подальшої інтеграції в застосунок для класифікації зображень.

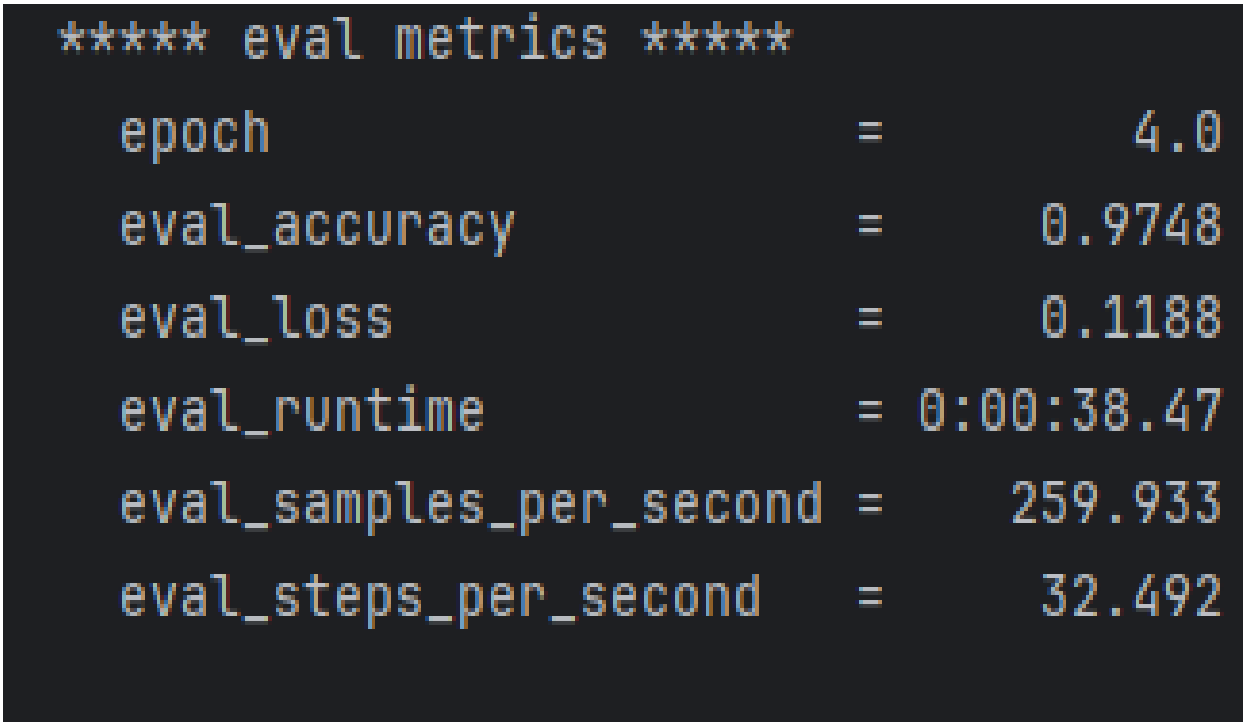

Рисунок 3.6 – Метрики точності моделі після навчання

Крок 3. Збереження результатів навчання моделі. Після успішного навчання модель зберігається для подальшого використання в застосунку. Цей етап є критично важливим, оскільки дозволяє використовувати натреновану модель для класифікації зображень у різних середовищах без необхідності повторного навчання.

Крок 4. Реєстрація Discord-бота через Discord Developer Portal. Створюється новий застосунок та додається бот до цього застосунку, налаштування необхідних дозволів та отримання токену автентифікації, який забезпечить безпечне підключення бота до сервера Discord (рис. 3.7). Важливо ретельно налаштувати права доступу, щоб бот мав лише ті можливості, які необхідні для виконання його функцій, що мінімізує ризики безпеки.

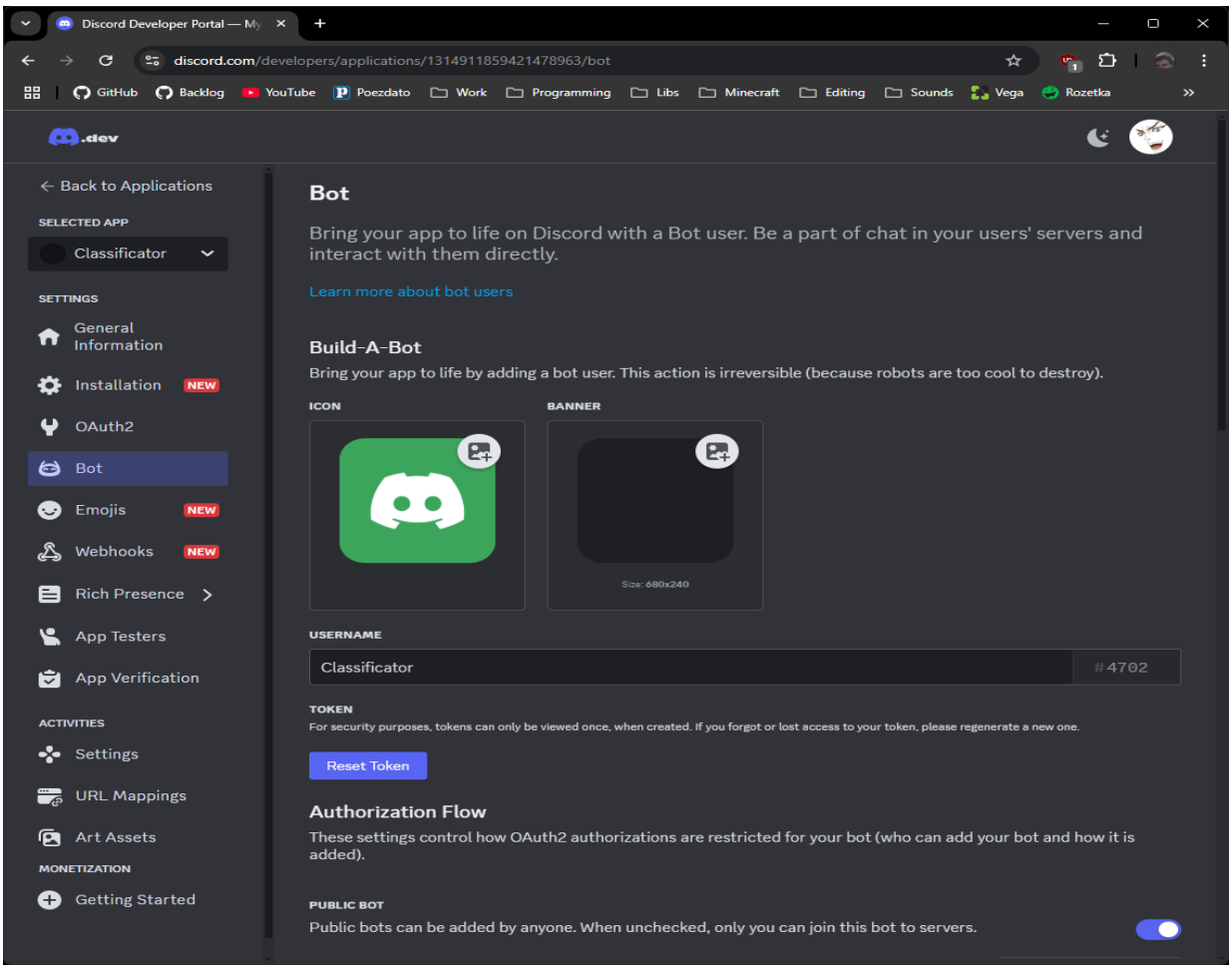

Рисунок 3.7 – Меню налаштування бота для Discord

Крок 5. Розробка основної логіки бота. Цей крок включає ініціалізацію бота з відповідним префіксом команд, налаштування інтентів (intents) для прослуховування потрібних подій, таких як отримання повідомлень та завантаження файлів. Реалізація обробки команд, наприклад, команди для класифікації зображень, дозволяє користувачам легко взаємодіяти з ботом. Важливо забезпечити коректну обробку вхідних даних, включаючи перевірку форматів зображень та їх попередню обробку перед подачею у модель для класифікації.

Крок 6. Інтеграція моделі ViT з ботом. Завантажуємо навчену модель та її оптимізацію для швидкої обробки запитів та забезпечення ефективної взаємодії з користувачами. Модель повинна бути здатною обробляти зображення в реальному часі, надаючи точні результати класифікації у відповідь на запити користувачів. Важливо також реалізувати механізми

обробки помилок, щоб бот міг коректно реагувати на некоректні запити або несподівані ситуації, забезпечуючи стабільну роботу застосунку.

Крок 7. Тестування бота. Цей крок включає перевірку всіх функцій, тестування взаємодії з різними типами зображень, а також оцінку продуктивності моделі. Виявлення та виправлення помилок на цьому етапі дозволяє забезпечити надійність та ефективність бота.

3.4 Інструкція користувача

Для того, щоб почати користуватися ботом, необхідно спочатку зареєструватися в застосунку Discord. Для цього необхідно перейти на офіційний сайт та обрати, яка версія буде зручніше: веб чи десктоп. Якщо користувач обрав вебверсію, то йому необхідно відкрити браузер і зайти на сайт Discord. Якщо користувачу більше підходить десктоп версія, то необхідно натиснути кнопку «Download» і після переходу на нову вкладинку натиснути кнопку «Завантажити для Windows» (рис. 3.8) та встановити програму, дотримуючись інструкцій на екрані.

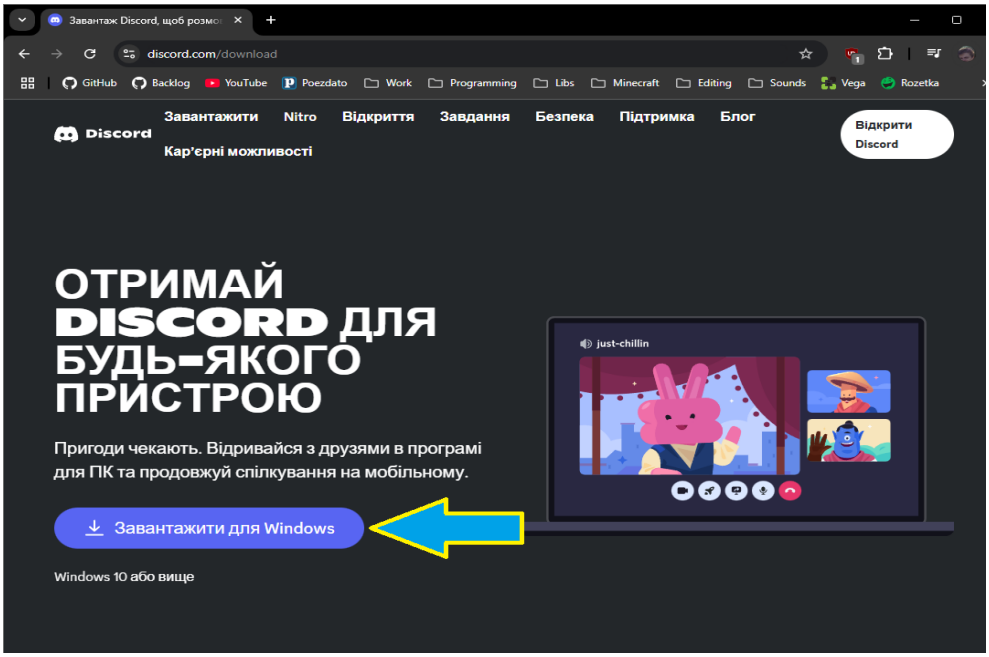

Рисунок 3.8 – Сторінка завантаження застосунку Discord

Після встановлення Discord необхідно відкрити застосунок і натиснути «Зареєструватися» (рис. 3.9). Необхідно заповнити реєстраційну форму, ввівши електронну пошту, створивши ім’я користувача та надійний пароль. Необхідно пройти перевірку, що користувач не є роботом, і погодитися з умовами використання сервісу. Після цього Discord надішле лист із посиланням для підтвердження електронної пошти. Необхідно відкрити цей лист і натиснути на посилання для активації облікового запису.

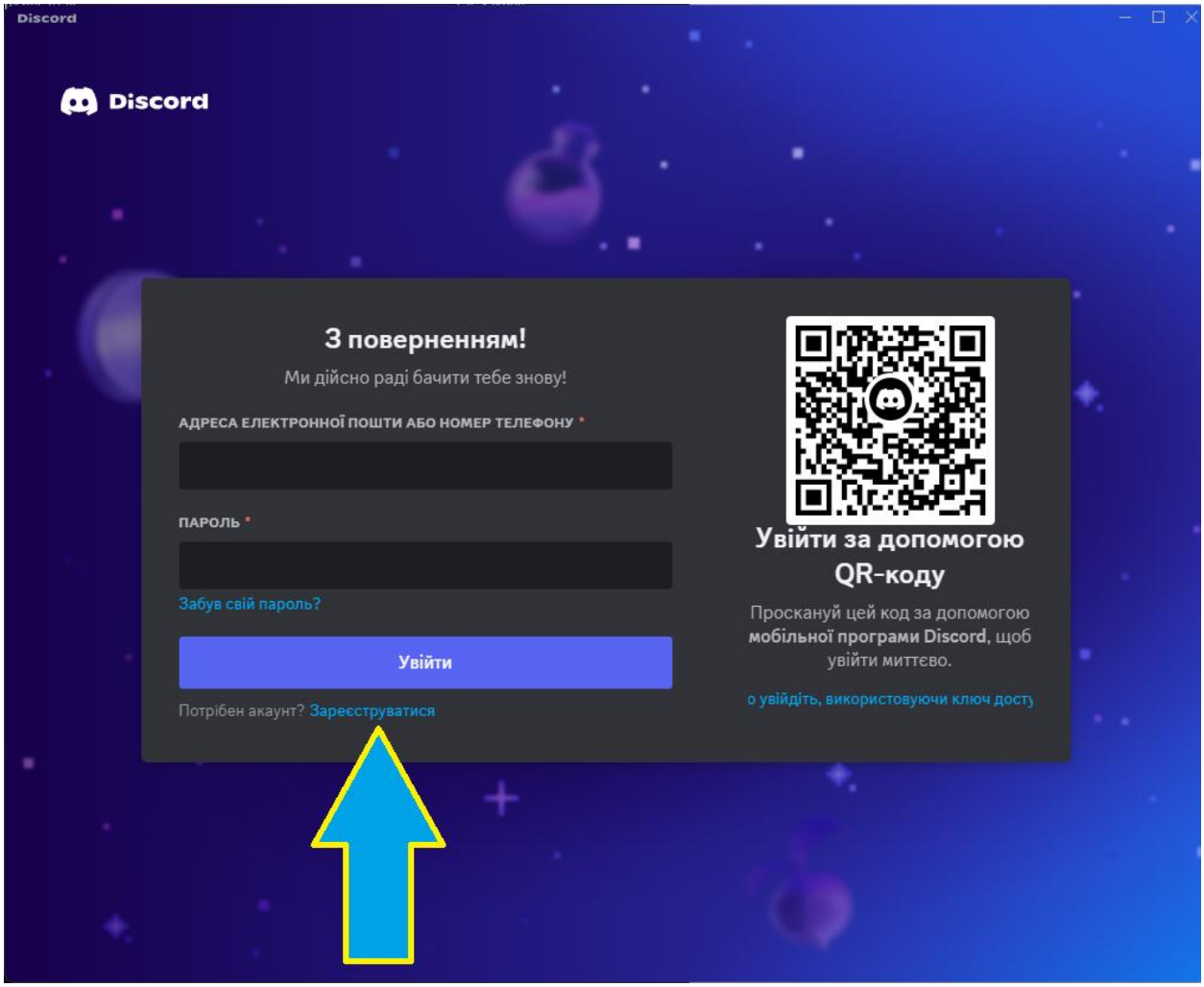

Рисунок 3.9 – Стартове вікно Discord

Коли обліковий запис буде активовано, необхідно зайти у Discord, використовуючи облікові дані. Якщо було обрано десктоп версію, просто треба відкрити його; для вебверсії – необхідно зайти на сайт і увійти в акаунт. Після входу з’явиться основний інтерфейс Discord, де зліва знаходяться

сервери, до яких користувач приєднаний, а справа – список друзів та активних чатів.

Для того, щоб взаємодіяти з чат-ботом, необхідно приєднатися до відповідного сервера, де цей бот присутній або створити власний сервер та додати на нього бота (рис. 3.10). Необхідно отримати запрошення на сервер, яке може бути отримане різними способами. Після чого необхідно натиснути на посилання-запрошення і підтвердити приєднання до серверу.

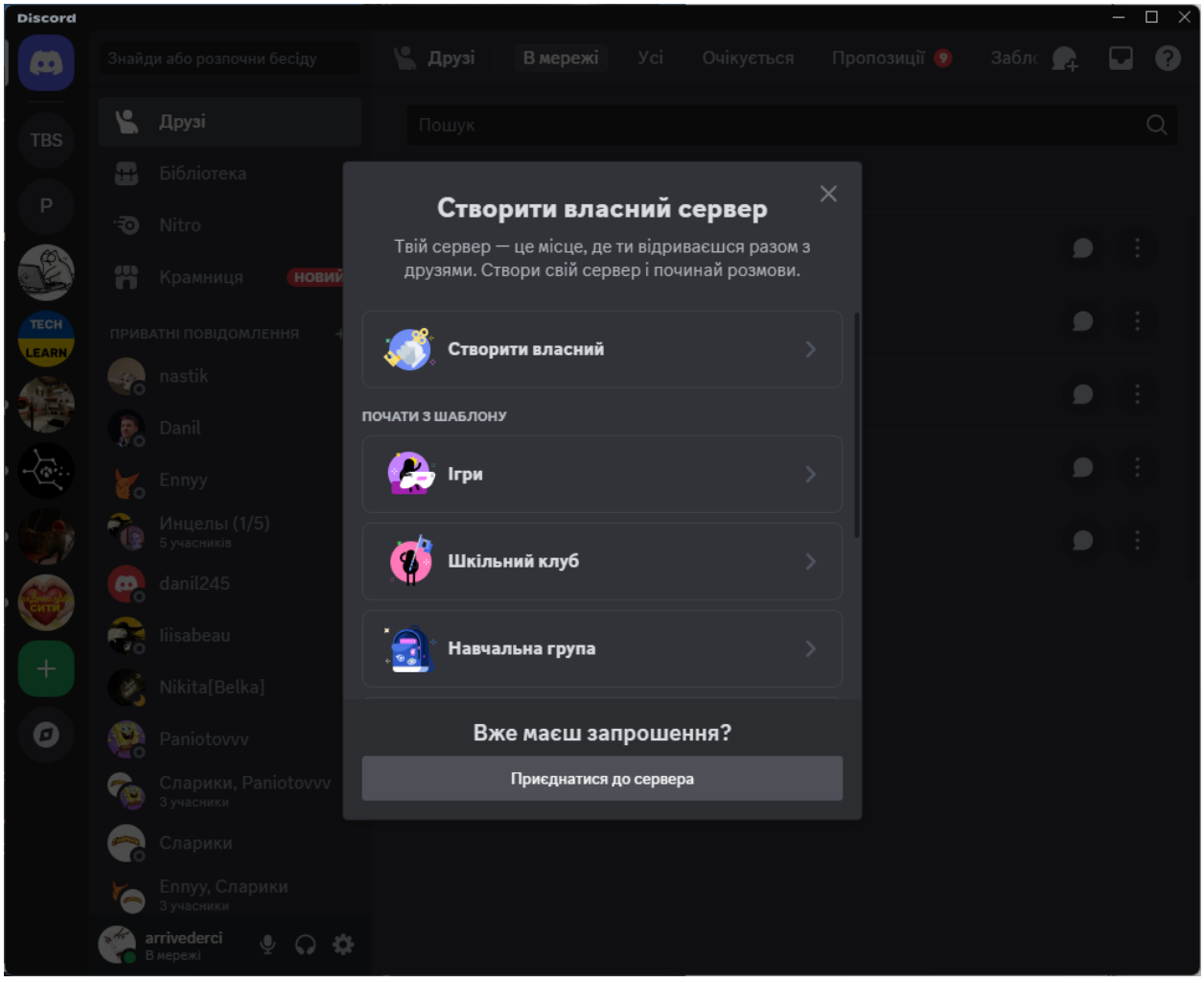

Рисунок 3.10 – Вікно вибору щодо приєднання до вже готового серверу чи створення власного

Для додавання бота на власний сервер необхідно активувати його посилання: https://discord.com/oauth2/authorize?client_id=1314911859421478963&permissions=8&integration_type=0&scope=bot. Після чого необхідно

натиснути кнопку «Продовжити» (рис. 3.11). Користувач обов’язково повинен мати права адміністратора на сервері.

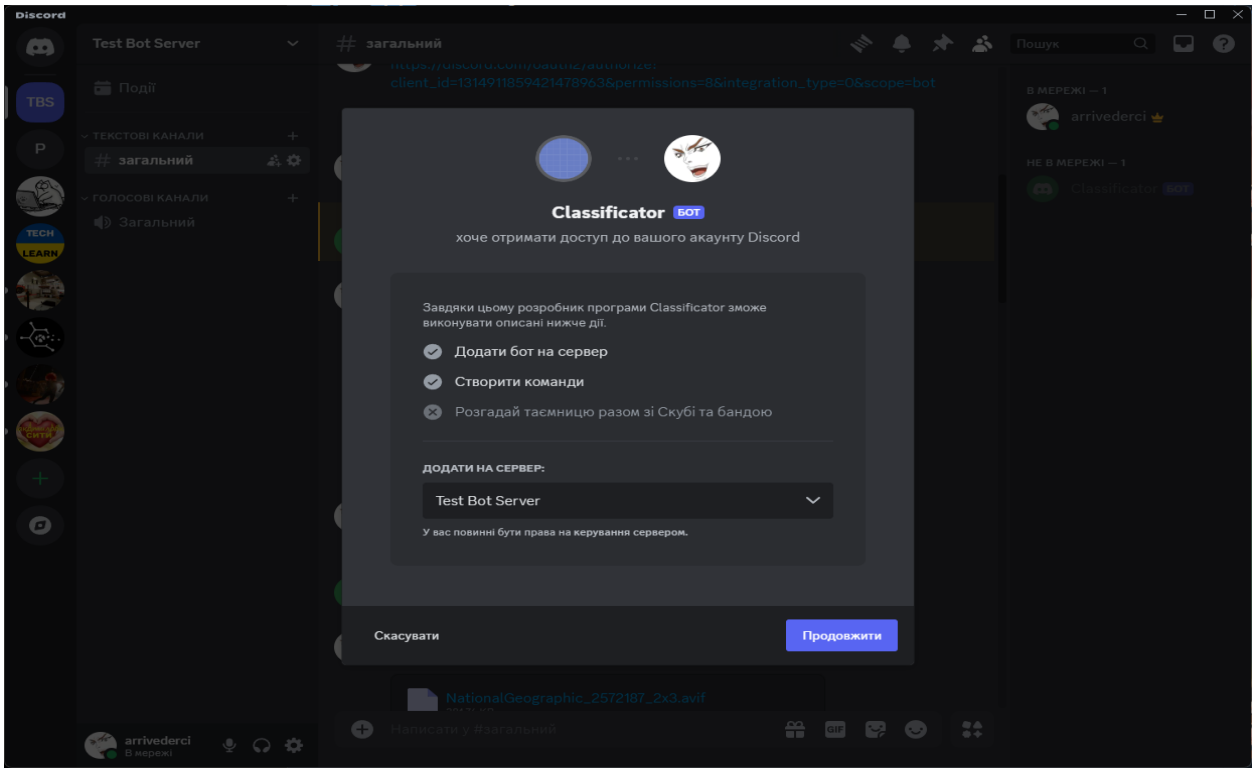

Рисунок 3.11 – Вікно додавання бота до серверу

Після приєднання до серверу треба перейти у текстовий канал, де дозволено використовувати бота та відправити необхідне зображення у текстовий чат. Необхідно натиснути на кнопку відправки файлу (рис. 3.12) і обрати «Завантажити файл на сервер».

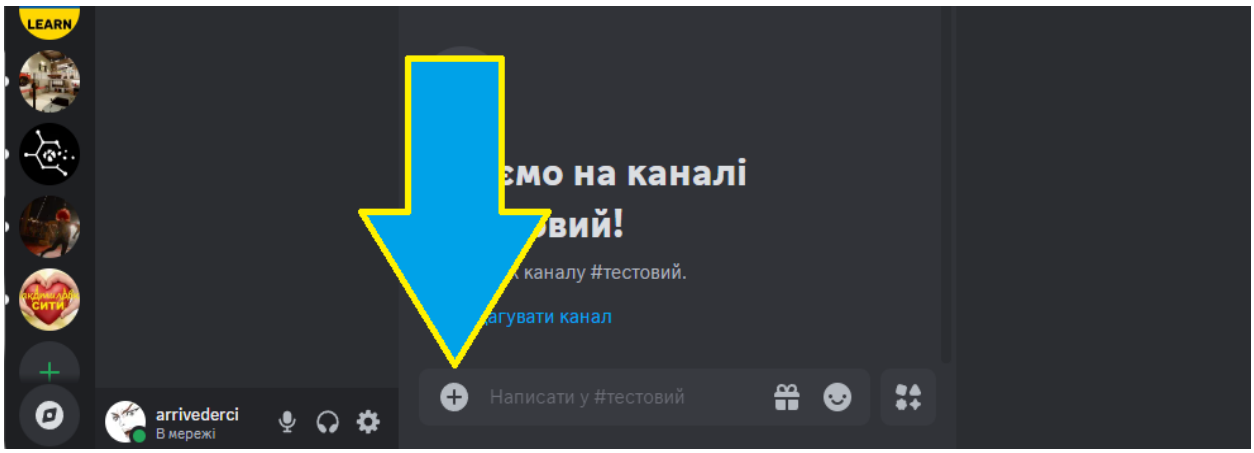

Рисунок 3.12 – Кнопка завантаження файлу до текстового чату

Після цього з’явиться вікно з вибором зображень для завантаження у застосунок (рис. 3.13).

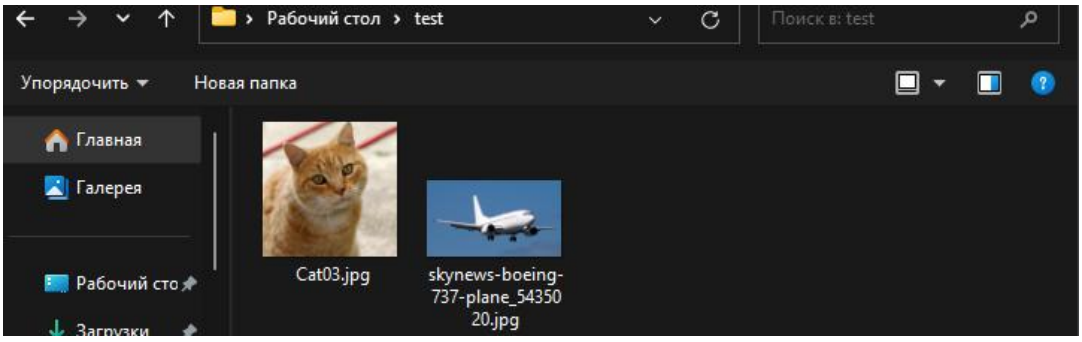

Рисунок 3.13 – Вікно вибору файлу

Після цього треба підтвердити завантаження файлу, натиснувши кнопку «Відкрити». Після завантаження зображення користувач побачить його у невеликому віконці для додаткових налаштувань (наприклад, можна замінити зображення у випадку помилкового завантаження). І нарешті необхідно натиснути клавішу «Enter», щоб фото відправилося і бот провів класифікацію зображення (рис. 3.14).

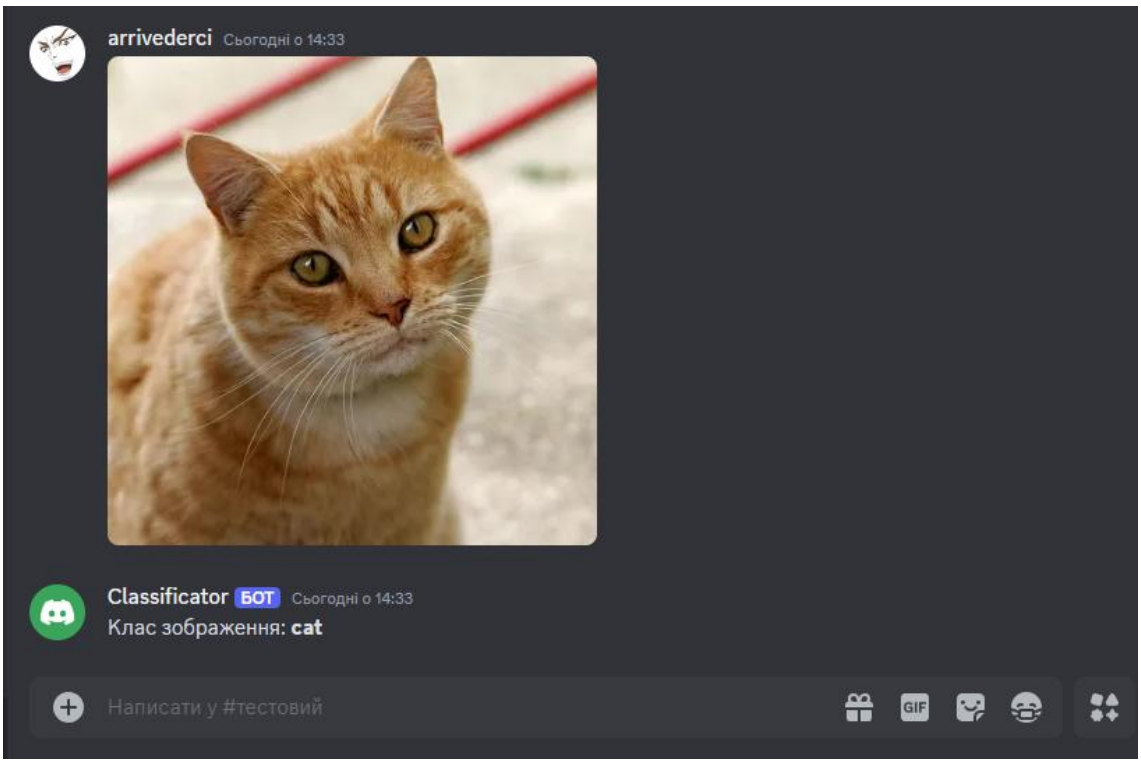

Рисунок 3.14 – Результат класифікації моделлю ViT

3.5 Тестування розробленої моделі

Тестування застосунків, пов’язаних з класифікацією зображень, є критично важливим етапом розробки, який забезпечує високу якість, надійність та ефективність функціонування системи. Тестування дозволяє виявити помилки та недоліки на ранніх стадіях розробки, що значно знижує витрати на їх виправлення в майбутньому. Це особливо важливо у застосунках, де точність класифікації має вирішальне значення, наприклад, у медичних діагностиках, де помилка може мати серйозні наслідки для здоров’я пацієнта. Крім того, тестування допомагає перевірити якість навчання моделі, забезпечуючи, що вона адекватно реагує на різноманітні та складні зображення, які можуть зустрічатися в реальних умовах експлуатації.

Детальне тестування також включає перевірку продуктивності алгоритмів класифікації під навантаженням, що гарантує швидку та ефективну обробку великої кількості зображень без затримок або збоїв у роботі застосунку. Це особливо важливо для застосунків, що обслуговують велику кількість користувачів або працюють у реальному часі, наприклад, у системах відеоспостереження чи автономних транспортних засобах. Тестування також охоплює перевірку сумісності застосунку з різними пристроями та операційними системами, що забезпечує його доступність для ширшої аудиторії користувачів.

Безпека є ще одним важливим аспектом, який охоплюється під час тестування. Захист даних, особливо конфіденційних зображень, від несанкціонованого доступу та забезпечення цілісності результатів класифікації, є критично важливими для збереження довіри користувачів та відповідності нормативним вимогам. Тестування також включає оцінку масштабованості системи, що дозволяє застосунку ефективно рости та адаптуватися до збільшення обсягу даних та кількості користувачів без втрати продуктивності.

Користувацький інтерфейс та взаємодія також проходять ретельне тестування, щоб забезпечити інтуїтивність та зручність використання застосунку. Це включає перевірку зрозумілості навігації, реакції на дії користувача та загального досвіду, що впливає на задоволеність кінцевих користувачів. Важливо також враховувати різні сценарії використання та можливі виклики, з якими можуть стикнутися користувачі, щоб забезпечити безперебійну та ефективну роботу застосунку у будь-яких умовах.

Автоматизація процесу тестування грає ключову роль у забезпеченні регулярної перевірки функціональності та стабільності застосунку, особливо при впровадженні нових функцій або оновлень алгоритмів класифікації. Це дозволяє швидко виявляти та виправляти помилки, підтримуючи високу якість продукту на всіх етапах його життєвого циклу. Крім того, автоматизовані тести сприяють стандартизації процесів тестування, що забезпечує послідовність та надійність результатів.

Загалом, тестування у застосунках для класифікації зображень є багатогранним процесом, який включає в себе перевірку точності алгоритмів, продуктивності, безпеки, сумісності та користувацького досвіду. Воно забезпечує високу якість кінцевого продукту, знижує ризики виникнення помилок, підвищує довіру користувачів та сприяє успішній інтеграції застосунку на ринку. Завдяки всебічному тестуванню, застосунок для класифікації зображень може ефективно виконувати свої функції, адаптуватися до змінних умов та вимог, забезпечуючи стабільну та надійну роботу протягом тривалого часу.

3.5.1 Причини проведення тестування

Тестування здійснюється зважаючи на причини, що сприяють підвищенню якості та надійності програмного забезпечення. До основних причин тестування належать:

– виявлення дефектів, що сприятиме виявленню програмних помилок, недоліків та дефектів у застосунку, що дозволяє розробникам усунути їх перед випуском продукту на ринок;

– перевірка точності класифікації. Тестування включає оцінку точності моделі класифікації зображень, щоб забезпечити відповідність результатів вимогам та очікуванням користувачів;

– оцінка інтеграції дозволить перевірити, як застосунок взаємодіє з іншими системами, платформами або пристроями, наприклад, з серверами, базами даних чи зовнішніми сервісами, забезпечуючи коректну інтеграцію;

– забезпечення стабільності та надійності визначить рівень стабільності та надійності застосунку, включаючи його здатність обробляти помилки та відновлюватися після збоїв;

– оптимізація продуктивності моделі допоможе виявити можливості для покращення швидкодії та ефективності.

Враховуючи ці аспекти, тестування застосунку з моделлю для класифікації зображень є важливою складовою процесу розробки програмного забезпечення, бо воно забезпечить високу якість продукту та задоволеність користувачів.

3.5.2 Основні етапи тестування

Початок тестування нейромережі для класифікації зображень вимагає ретельного планування та підготовки. Спершу необхідно детально вивчити вимоги до застосунку, що допоможе зрозуміти його функціонування та очікування користувачів. Вимоги можуть бути зафіксовані в специфікаціях, технічній документації або сформульовані у вигляді завдань.

Після аналізу вимог слід створити план тестування, який охоплює ключові аспекти функціональності та сценарії використання застосунку. Вибір методів тестування має відповідати специфіці застосунку та особливостям

моделі класифікації зображень. Наступним кроком є розробка детальних тестових сценаріїв, що включають послідовність кроків для перевірки різних функцій та можливостей застосунку. Це забезпечує систематичний підхід до тестування різних компонентів та гарантує, що жоден аспект не буде пропущений.

Далі необхідно підготувати тестове середовище, яке включає необхідне обладнання, програмне забезпечення та тестові дані. Важливо переконатися, що всі ресурси доступні та налаштовані для проведення тестів без перешкод. Підготовка середовища також включає інтеграцію нейромережі класифікації зображень з іншими компонентами застосунку, щоб забезпечити їхню взаємодію у реалістичних умовах.

Після підготовки середовища можна приступати до виконання тестових сценаріїв. Це передбачає поетапне виконання кожного кроку тесту та перевірку отриманих результатів відповідно до очікуваних. Під час цього процесу необхідно ретельно документувати всі виявлені помилки, дефекти або неполадки, що виникають у застосунку або моделі класифікації.

На завершальному етапі проводиться оцінка результатів тестування. Аналізуються виявлені проблеми та помилки, після чого формується звіт про тестування, який містить детальну інформацію про знайдені дефекти, їх пріоритети та рекомендації щодо виправлення. Після виправлення виявлених недоліків проводиться повторне тестування, щоб переконатися, що виправлення були успішними і не спричинили появу нових дефектів.

Процес тестування може змінюватися в залежності від специфіки проєкту та обраної методології розробки. Важливо також враховувати тестування на ранніх етапах розробки та використовувати автоматизовані інструменти для тестування, де це можливо. Це дозволяє підвищити ефективність тестування, забезпечити його повторюваність та знизити ризик помилок.

3.5.3 Тестування застосунку

Тестування застосунку слід розпочати з відкриття Discord. Після запуску застосунку можна помітити, що бот, відповідальний за класифікацію зображень, вимкнений (рис. 3.15). Це означає, що для коректної роботи нейромережі необхідно активувати бота на серверній частині. Для розміщення програми, яка відповідатиме за бота, існує кілька надійних серверних платформ, як от Amazon, Google Cloud Platform, Microsoft Azure тощо.

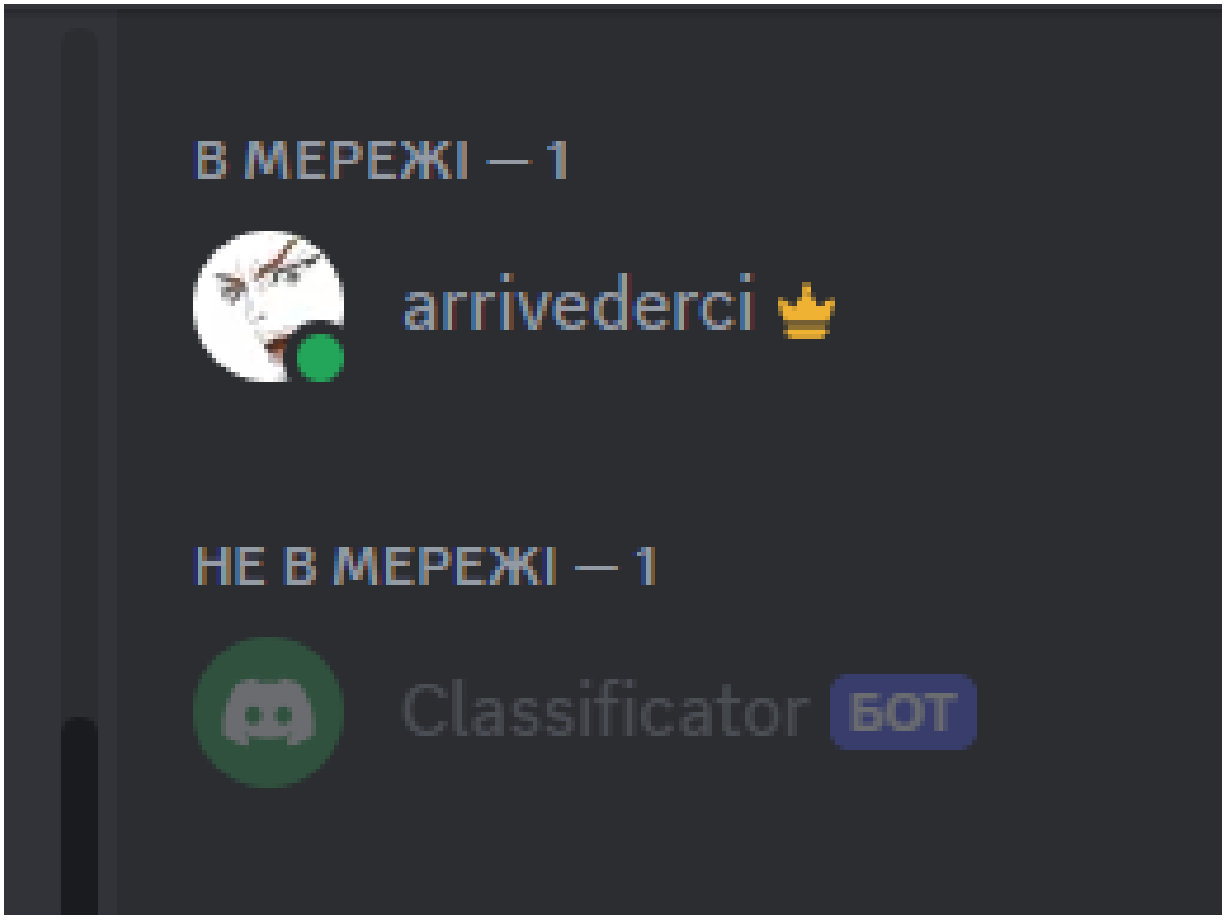

Рисунок 3.15 – Класифікатор не спрацює через те, що бот вимкнений

Після запуску серверної частини необхідно переконатися, що нейромережа правильно налаштована і підключена до бота. Для цього необхідно завантажити тестове зображення автомобіля (рис. А.1) та собаки (рис. А.2) для виконання кількох тестових запитів (рис. 3.16), щоб перевірити, чи відповідає класифікація очікуваним результатам.

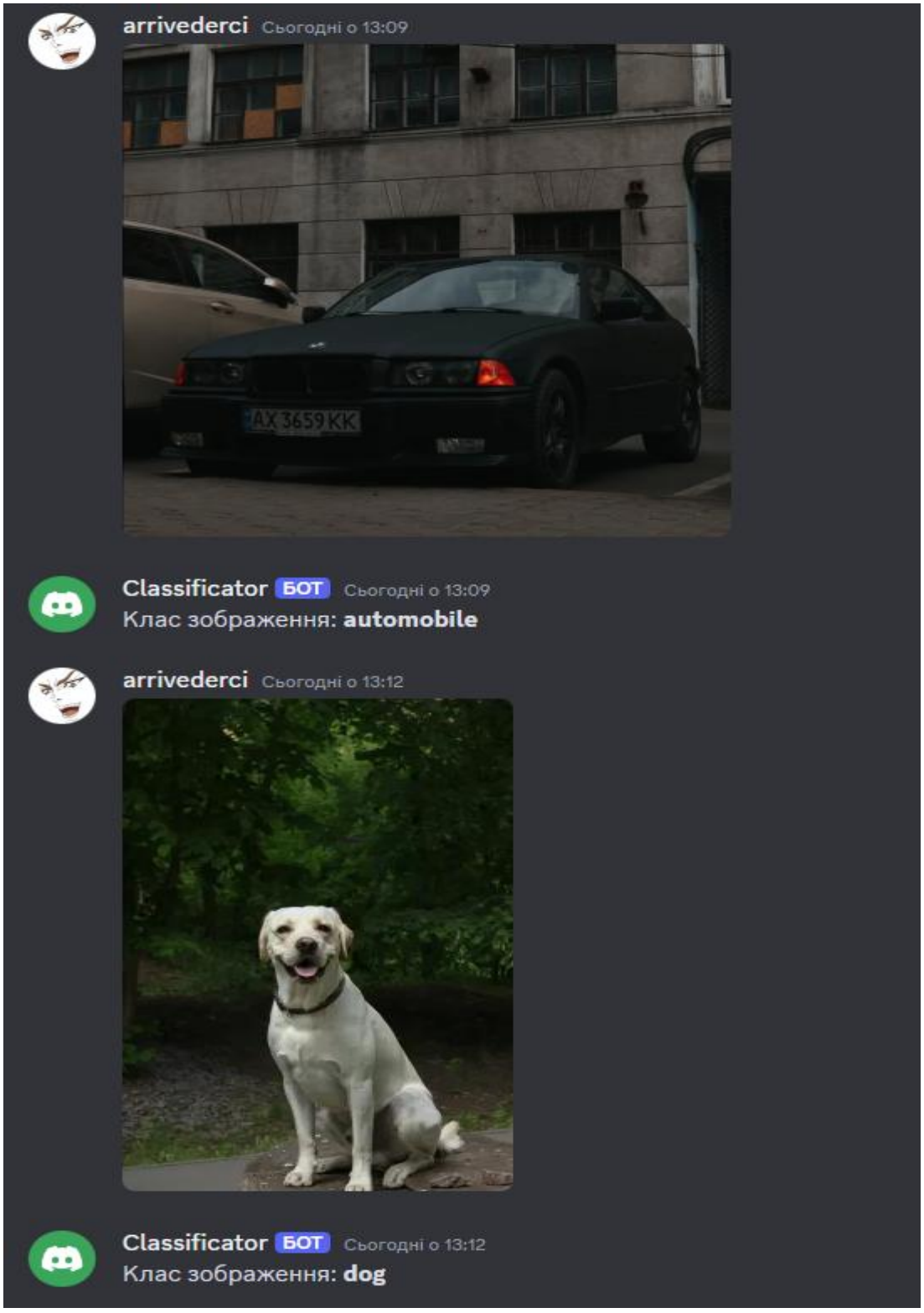

Рисунок 3.16 – Кілька тестових запитів до бота для перевірки працездатності

Після перевірки працездатності і коректності роботи нейромережі, необхідно перевірити як вона буде поводитися в різних ситуаціях.

Слід протестувати нейромережу у ситуації, коли користувач завантажив більше одного зображення (рис. 3.17).

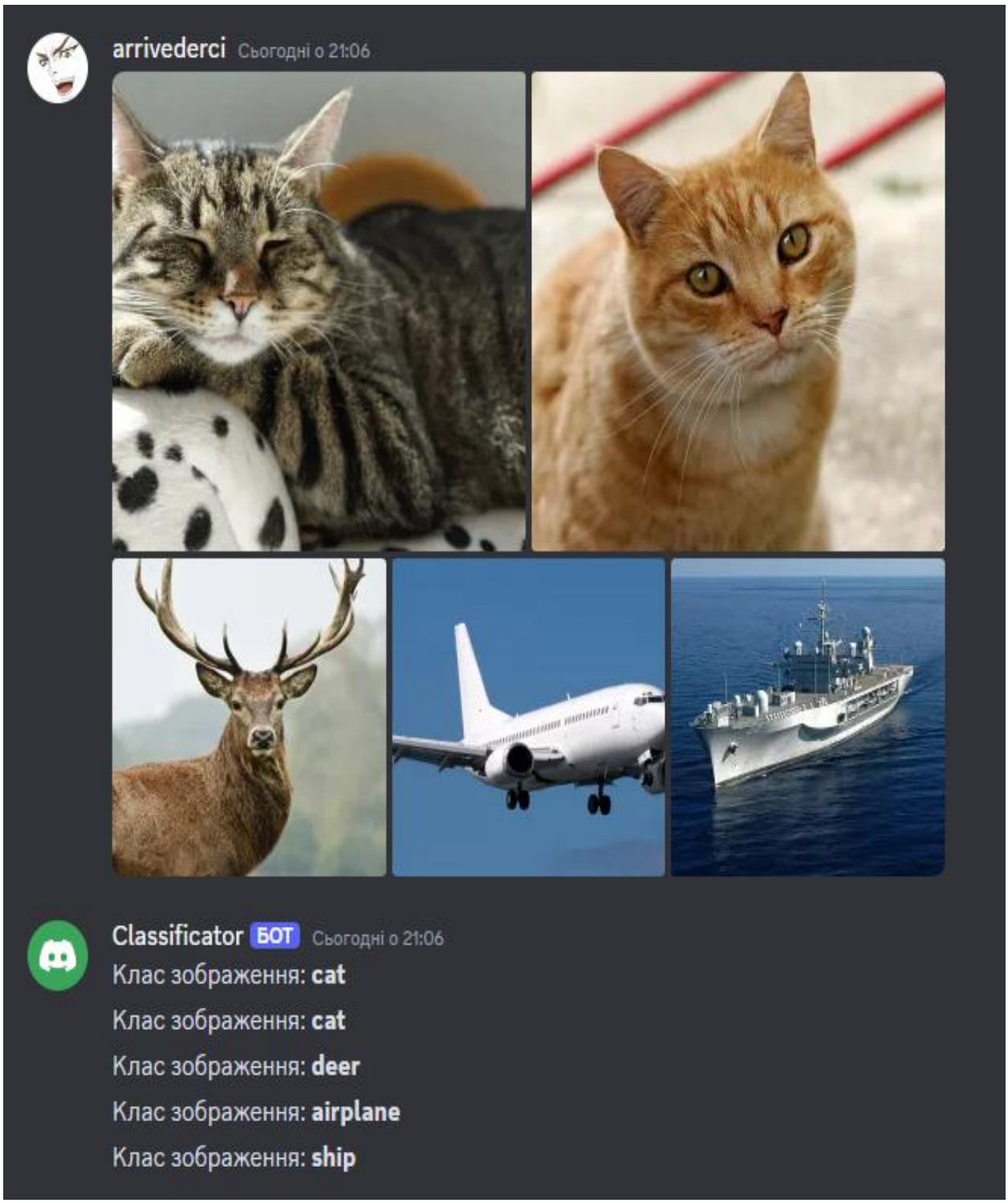

Рисунок 3.17 – Користувач завантажує декілька зображень

Як можна помітити, нейромережа чудово класифікує групу зображень. Вона робить це відповідно до того, в якому порядку зображення були завантажені.

Ще треба протестувати ситуацію в якій користувач спробує завантажити велику кількість файлів (рис. 3.18).

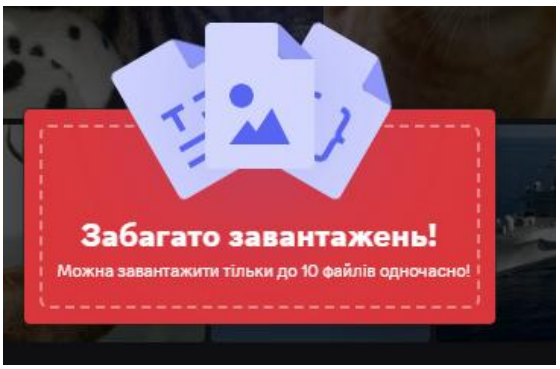

Рисунок 3.18 – Повідомлення про те, що користувач намагається завантажити багато зображень за раз

Це обмеження накладається зі сторони Discord задля того, щоб зберегти продуктивність та уникнути перевантаження застосунку, оскільки їм користуються мільйони людей по всьому світу одночасно.

Тепер необхідно перевірити як модель відпрацює, якщо користувач завантажить зображення, яке не входить у множину класів набору даних CIFAR-10 (рис. 3.19).

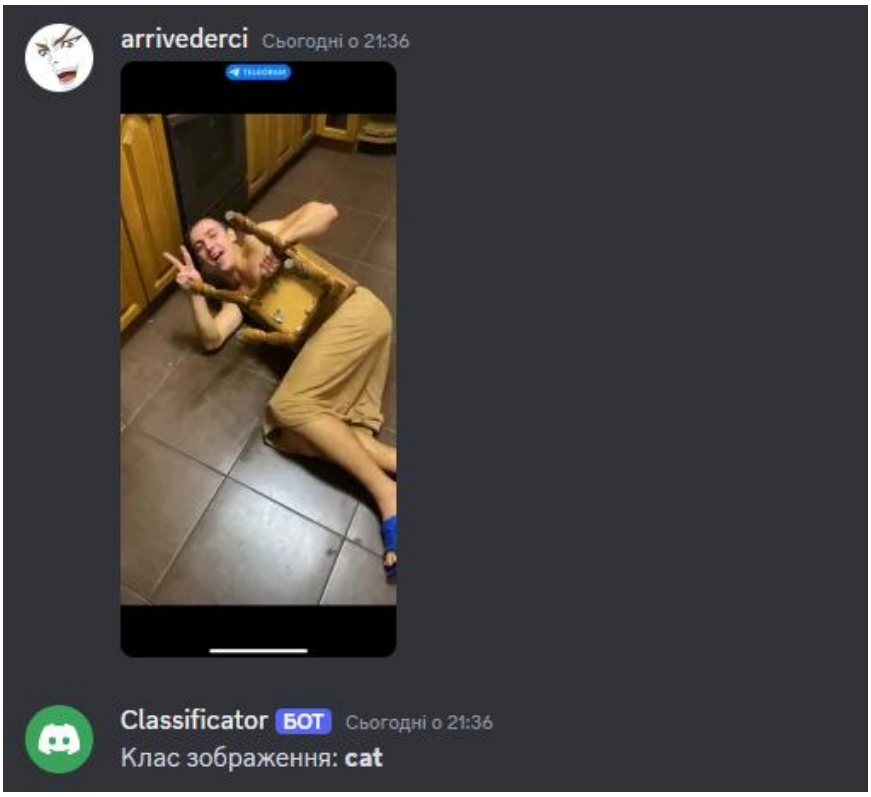

Рисунок 3.19 – Нейромережа неправильно класифікує через обмежену множину класів набору даних CIFAR-10

Оскільки набор даних CIFAR-10 містить лише 10 класів для навчання, а саме: «airplane», «automobile», «bird», «cat», «deer», «dog», «frog», «horse», «ship» та «truck», то для вирішення цієї проблеми можна розширення дані, що допоможе створити різноманітніші приклади для навчання і це дозволить моделі краще узагальнювати інформацію. Для подальшої роботи нейромережі можна використовувати інші набори даних, наприклад, CIFAR-100, ImageNet тощо, але для цього необхідно перенавчати нейромережу для більш якісної класифікації, а на це може піти у декілька разів більше часу та ресурсів комп’ютера.

Тобто, провівши тестування моделі можна побачити, що нейромережа досить точно класифікує зображення, але ще більш точній класифікації може сприяти розширення навчального набору даних.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод класифікації зображень за допомогою Vision Transformer.

Проведено детальний аналіз традиційних методів класифікації зображень із використанням CNN та RNN. Розглянуто основні принципи їхньої роботи, сильні та слабкі сторони, а також актуальність використання в сучасних задачах.

Реалізовано нейромережу для класифікації зображень на основі Vision Transformer, що ґрунтується на механізмах уваги, які продемонстрували високу ефективність у задачах класифікації. Точність моделі на наборі даних CIFAR-10 складає приблизно 97,5%, що є досить високим показником.

Для полегшення взаємодії з реалізованою нейромережею створено Discord-бота, який виступає як зручний користувацький інтерфейс. Бот дозволяє завантажувати зображення, запускати процес класифікації та отримувати результати без необхідності використання складних програмних інструментів.

Наукова новизна роботи полягає у можливості використання Vision Transformer у задачах класифікації зображень у порівнянні із CNN, що відкриває нові перспективи для використання трансформерів у комп’ютерному зорі, які раніше застосовувались переважно у сфері обробки природної мови.

Результати дослідження апробовано у вигляді тез доповіді під час X Міжнародної науково-практичної конференції «PERSPECTIVES OF CONTEMPORARY SCIENCE: THEORY AND PRACTICE» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

2. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57-70.

3. Lyashenko, V., Kobylin, O., Ryazantsev, O., Ryazantsev, I., Barbaruk, V., & Zhychenko, Y. (2020). General Ideology of Analysis Digital Medical Images in RGB Format.

4. Гороховатський, В. О., Творошенко, І. С., & Чмутов, Ю. В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень.

5. Гороховатський, В. О., Творошенко, І. С., & Сидоренко, Д. (2021). Класифікація зображень із використанням кластерного подання.

6. Кобилін, О., Вечірська, І., & Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 3, pp. 63-76.

7. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, International Journal of Academic and Applied Research, 7(11), pp. 134-145.

8. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. *Information Technology: Computer Science, Software Engineering and Cyber Security*, 3, pp. 97–107.

9. Image Processing: How Do Image Classifiers Work. URL: https://levity.ai/blog/how-do-image-classifiers-

work#:~:text=CNNs%20are%20a%20special%20form,layers%20to%20generate%20a%20classification (дата звернення 08.10.2024).

10. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

11. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

12. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks. URL: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/ (дата звернення 08.10.2024).

13. How I Classified Images With Recurrent Neural Networks. URL: https://medium.com/@nathaliejeans/how-i-classified-images-with-recurrent-neural-networks-28eb4b57fc79 (дата звернення 08.10.2024).

14. Корякіна, С., & Творошенко, І. С. (2024). Особливості методу SIAMESE NETWORKS щодо його застосування до задачі класифікації об’єктів на зображеннях.

15. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

16. Використовуємо CNN для обробки зображень. Частина перша. URL: https://dou.ua/forums/topic/48368/ (дата звернення 14.10.2024).

17. Best Practices for Image Preprocessing in Image Classification. URL: https://keylabs.ai/blog/best-practices-for-image-preprocessing-in-image-classification/ (дата звернення 22.10.2024).

18. Object Localization and Image Localization. URL: https://viso.ai/computer-vision/object-localization-and-image-localization/ (дата звернення 22.10.2024).

19. Object Detection Part-1: Introduction to object detection. URL: https://medium.com/@kattarajesh2001/object-detection-part-1-introduction-to-object-detection-321f1fd56295 (дата звернення 22.10.2024).

20. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network. URL: https://www.upgrad.com/blog/basic-cnn-architecture/ (дата звернення 22.10.2024).

21. Convolutional Neural Network: Everything You Need to Know. URL: https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know (дата звернення 22.10.2024).

22. Convolutional Neural Networks: A Deep Dive into Architectures and Layers. URL: https://pareto.ai/blog/convolutional-neural-networks (дата звернення 22.10.2024).

23. Vision Transformer Explained | Papers With Code. URL: https://paperswithcode.com/method/vision-transformer (дата звернення 28.10.2024).

24. Vision Transformer (ViT). URL: https://huggingface.co/docs/transformers/model_doc/vit (дата звернення 28.10.2024).

25. Vision Transformer: What It Is & How It Works [2024 Guide]. URL: https://www.v7labs.com/blog/vision-transformer-guide (дата звернення 28.10.2024).

26. Vision Transformers (ViT) in Image Recognition – 2024 Guide. URL: https://viso.ai/deep-learning/vision-transformer-vit/ (дата звернення 28.10.2024).

27. Vision Transformers, Explained. URL: https://towardsdatascience.com/vision-transformers-explained-a9d07147e4c8 (дата звернення 28.10.2024).

28. Different Evaluation metrics for Computer Vision tasks. URL: https://medium.com/@nikitamalviya/different-evaluation-metrics-for-computer-vision-tasks-83feb9a1041b (дата звернення 30.10.2024).

29. Computer Vision Model Performance Evaluation (Guide 2024). URL: https://viso.ai/computer-vision/model-performance/ (дата звернення 30.10.2024).

30. A Gentle Introduction to Exploding Gradients in Neural Networks. URL: https://machinelearningmastery.com/exploding-gradients-in-neural-networks/ (дата звернення 02.11.2024).

31. PyCharm: the Python IDE for data science and web development. URL: https://www.jetbrains.com/pycharm/ (дата звернення 04.11.2024).

32. Our Documentation | Python.org. URL: https://www.python.org/doc/ (дата звернення 04.11.2024).

33. Project Jupyter | Home. URL: https://jupyter.org/ (дата звернення 04.11.2024).

34. Discord Developer Portal – Documentation – Intro. URL: https://discord.com/developers/docs/intro (дата звернення 24.11.2024).

35. PyTorch. URL: https://pytorch.org/ (дата звернення 30.11.2024).

36. torchvision – Torchvision 0.20 documentation. URL: https://pytorch.org/vision/stable/index.html (дата звернення 30.11.2024).

37. Datasets. URL: https://huggingface.co/docs/datasets/index (дата звернення 30.11.2024).

38. Transformers. URL: https://huggingface.co/docs/transformers/index (дата звернення 30.11.2024).

39. Welcome to discord.py. URL: https://discordpy.readthedocs.io/en/stable/ (дата звернення 30.11.2024).

40. Yakovenko V. (2024) Сучасні архітектури нейромереж для класифікації зображень. *Perspectives of contemporary science: theory and practice. Proceedings of the 10th International scientific and practical conference, 11-13.11.2024, Lviv, Ukraine*, pp. 453-459.