ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Телеграм-бот «AiPaws» для автоматичної класифікації зображень тварин з використанням нейронних мереж»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-21
Марко МАХИНЯ
Керівник: викладач кафедри інформаційного менеджменту, математики та статистики
Олег МУШИНСЬКИЙ

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2025 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри комп'ютерних наук

_____Сергій МІЧКІВСЬКИЙ
«____»_____20___р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Махиня Марко Ігорович

Тема роботи	Телеграм-бот «AiPaws» для автоматичної класифікації зображень тварин з використанням нейронних мереж
Номер та дата наказу про затвердження теми	№24-1 від 14 березня 2025 року
Коротка постановка завдання	Створити Telegram-бота «AiPaws», який на основі глибоких нейронних мереж зможе автоматично класифікувати зображення тварин, визначаючи їхній вид та основні характеристики
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Жадан, О. (2022). Класифікація зображень на основі багатокомпонентних моделей даних 2. Rianto, R., Rahmatulloh, A., & Firmansah, T. A. (2019). Telegram bot implementation in academic information services with the forward chaining method. Sinkron: jurnal dan penelitian teknik informatika, 3(2), 73-78.
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп'ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 6 березня 2025 р.

Керівник Олег МУШИНСЬКИЙ

Здобувач освітнього ступеня бакалавра Марко МАХИНЯ

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження	01.03.2025 р.	*виконано*
2	Формування теми та призначення керівника	05.03.2025 р.	*виконано*
3	Затвердження теми кваліфікаційної роботи	14.03.2025 р.	*виконано*
4	Затвердження завдання на кваліфікаційну роботу	14.03.2025 р.	*виконано*
Основний етап			
5	Розробка концепції кваліфікаційної роботи	15.03.2025 р.	*виконано*
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	20.03.2025 р.	*виконано*
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	23.03.2025 р.	*виконано*
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	24.03.2025 р.	*виконано*
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	31.03-04.04.2025 р.	*виконано*
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	07.04.2025 р.	*виконано*
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	14.04.2025 р.	*виконано*
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	21.04.2025 р.	*виконано*
Завершальний етап			
13	Представлення рукопису для перевірки на плагіат	28.04-04.05.2025 р.	*виконано*
14	Підготовка презентації та доповіді на передзахист	05.05-11.05.2025 р.	*виконано*
15	Передзахист кваліфікаційної роботи	12.05-16.05.2025 р.	*виконано*
16	Доопрацювання роботи за результатами передзахисту	19.05-06.06.2025 р.	*виконано*
17	Експертиза роботи керівником та зовнішнім експертом	09.06-15.06.2025 р.	*виконано*
18	Доопрацювання доповіді та презентації для захисту	09.06-15.06.2025 р.	*виконано*
19	Захист кваліфікаційної роботи	16.06-22.06.2025 р.	*виконано*

Керівник Олег МУШИНСЬКИЙ

Здобувач освітнього ступеня бакалавра Марко МАХИНЯ

Махиня М.І. Телеграм-бот для автоматичної класифікації зображень тварин з використанням нейронних мереж

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма - Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2025.

Описано розробку Telegram-бота, який на основі глибоких нейронних мереж зможе автоматично класифікувати зображення тварин, визначаючи їхній вид та основні характеристики.

Ключові слова: телеграм-бот, нейронна мережа, комп'ютерний зір, задача класифікації.

Makhynya M.I. Telegram-bot for automatic classification of animal images using neural networks

Project explanatory note by specialty 122 – Computer science. – «KROK» University, Educational and Scientific Institute of Information and communication technologies, Department of Computer Science, Kyiv, 2025.

The development of a Telegram-bot is described, which, based on deep neural networks, will be able to automatically classify images of animals, determining their species and main characteristics.

Keywords: telegram bot, neural network, computer vision, classification problem.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі нейронні мережі та технології штучного інтелекту відіграють важливу роль у багатьох сферах життя, зокрема у сфері обробки та аналізу зображень. Однією з важливих задач комп'ютерного зору є автоматична класифікація зображень, що знаходить застосування у ветеринарії, наукових дослідженнях, зоології, екологічному моніторингу та багатьох інших галузях. Використання Telegram-ботів для вирішення таких завдань є перспективним напрямом, оскільки вони забезпечують зручний доступ до функціоналу без необхідності встановлення додаткових програм. У зв'язку з цим розробка Telegram-бота для автоматичної класифікації зображень тварин із використанням нейронних мереж є актуальною задачею, що поєднує сучасні технології штучного інтелекту з практичними потребами користувачів.

Мета дослідження. Метою дослідження є розробка Telegram-бота, який забезпечує автоматичну класифікацію зображень тварин із використанням нейронних мереж, що дозволяє покращити точність та швидкість ідентифікації видів тварин.

Для досягнення мети проєкту були виконані наступні завдання:

- аналіз існуючих підходів до класифікації зображень тварин;
- вибір та навчання нейронної мережі для обробки зображень;
- розробка та реалізація Telegram-бота для інтеграції моделі класифікації;
- тестування роботи бота та оцінка його ефективності.

Об'єктом дослідження є процес автоматичної класифікації зображень тварин із використанням технологій машинного навчання та комп'ютерного зору.

Предметом дослідження є алгоритми та методи нейронних мереж, що використовуються для класифікації зображень тварин, а також принципи розробки Telegram-ботів для інтеграції цих алгоритмів у користувацьке середовище.

Методи дослідження. У процесі дослідження застосовуються такі методи:

- аналіз наукової літератури та існуючих підходів до класифікації зображень тварин;
- використання методів глибокого навчання для побудови та навчання нейронної мережі;
- програмна реалізація Telegram-бота та інтеграція моделі машинного навчання;
- тестування та аналіз точності роботи розробленого бота.

Практичне значення. Результати роботи можуть бути використані для автоматизації процесу ідентифікації тварин у наукових та освітніх установах, ветеринарних клініках, природоохоронних організаціях, а також у сфері розваг та освіти. Розроблений Telegram-бот забезпечує зручний доступ до інструментів комп'ютерного зору, що дозволяє значно спростити процес класифікації зображень тварин і зробити його доступним для широкого кола користувачів.

Структура та обсяг пояснювальної записки. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (25 найменувань). Пояснювальна записка містить 18 рисунків. Загальний обсяг пояснювальної записки складає 52 сторінки, основний зміст викладено на 46 сторінках

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ТЕЛЕГРАМ-БОТА «AIPAWS» ДЛЯ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ТВАРИН З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

1.1 Предметна область

Автоматична класифікація зображень є однією з найважливіших задач комп'ютерного зору і має широкий спектр застосувань у прикладних задачах, зокрема у біології, ветеринарії, екології, природоохоронних заходах, а також у розважальних та освітніх проєктах. Важливою частиною цього напряму є ідентифікація видів тварин на основі зображень, що дозволяє покращити процеси моніторингу біорізноманіття, спостереження за рідкісними видами та надання допомоги у ветеринарних дослідженнях.

Комп'ютерний зір є галуззю штучного інтелекту, що займається розробкою алгоритмів і моделей, які дозволяють комп'ютерам аналізувати та інтерпретувати візуальні дані. В останні десятиліття значного розвитку досягли методи глибокого навчання, які засновані на використанні штучних нейронних мереж. Глибокі нейронні мережі, зокрема згорткові нейронні мережі (Convolutional Neural Networks, CNN), демонструють високу ефективність у задачах класифікації зображень.

Задля класифікації зображень тварин застосовуються наступні архітектури нейронних мереж:

– ResNet (Residual Networks) – архітектура для вирішення проблем зникнення градієнтів у нейромережах. Відрізняється високою точністю класифікації.

– EfficientNet – оптимізує співвідношення продуктивності та швидкодії. Використовується переважно у мобільних додатках та ботах.

– MobileNet – полегшена модель, оптимізована для роботи на мобільних пристроях та веб-сервісах.

Telegram-бот, що працює із системами машинного навчання, може ефективно інтегрувати класифікаційні моделі, обробляючи зображення користувачів і надаючи їм результат у зручному форматі.

Telegram-боти є популярним інструментом для автоматизації багатьох задач, зокрема у сфері штучного інтелекту. Вони мають такі переваги:

– Легкість доступу – не потребують встановлення окремих застосунків, достатньо мати Telegram.

– Автоматизація процесів – користувачі можуть швидко отримувати відповіді на запити без необхідності самостійно шукати інформацію.

– Масштабованість – можливість обробки великої кількості запитів одночасно.

Telegram-боти, які працюють із технологіями машинного навчання, можуть приймати зображення від користувачів, аналізувати їх за допомогою нейронних мереж і повертати результати класифікації разом із додатковою інформацією про розпізнану тварину. Це дозволяє використовувати бота в освітніх та наукових цілях, а також для підвищення обізнаності про тваринний світ.

Незважаючи на успіхи комп’ютерного зору, існують низка проблем, пов’язаних із автоматичною класифікацією тварин:

– варіативність зображень – тварини можуть мати різні пози, освітлення та фонове середовище, що ускладнює процес розпізнавання;

– схожість між видами – деякі тварини мають схожі візуальні характеристики, що може спричиняти помилки класифікації;

– обмеженість навчальних даних – не всі види представлені у великих датасетах, що може впливати на якість навчання моделей;

– обробка в реальному часі – ефективність роботи моделі повинна дозволяти отримувати результати у мінімальний проміжок часу.

Telegram-бот буде вирішувати ці проблеми шляхом використання високоточних нейронних мереж, спеціальних методів аугментації даних та ефективної обробки запитів у реальному часі.

Предметна область дослідження охоплює використання нейронних мереж для автоматичної класифікації зображень тварин. Telegram-бот є ефективним інструментом для інтеграції алгоритмів комп'ютерного зору у взаємодію з користувачами. Сучасні моделі глибокого навчання, такі як ResNet, EfficientNet та MobileNet, дозволяють досягти високої точності класифікації. Водночас існують певні виклики у цьому напрямі, які потребують вирішення шляхом оптимізації моделей, покращення навчальних даних та забезпечення швидкої обробки запитів.

1.2 Огляд аналогів

Задача автоматичної класифікації зображень тварин є активно досліджуваною та реалізованою у різноманітних проектах. Існує кілька відомих рішень, які застосовують штучний інтелект та машинне навчання для автоматичного розпізнавання тварин на зображеннях. Огляд основних аналогів дозволяє оцінити рівень розвитку цієї технології та виявити можливості для покращення у розробці Telegram-бота.

Google Lens (рис 1.1.) є одним із найпопулярніших сервісів для ідентифікації об'єктів на зображеннях, включаючи тварин. Це інструмент на базі глибокого навчання, який дозволяє користувачам завантажувати зображення та отримувати відповідні результати розпізнавання. Google Lens використовує алгоритми комп'ютерного зору, щоб визначити види тварин, рослин та об'єктів, а також надати корисну додаткову інформацію, таку як місце проживання тварини, її характеристики та особливості.

Переваги Google Lens:

– велика база даних для навчання моделей;

– підтримка різних типів пристроїв (мобільні телефони, планшети);

– швидке розпізнавання та висока точність у більшості випадків.

Недоліки:

– можливі помилки класифікації через схожість між видами;

– не завжди доступна детальна інформація по менш популярним або рідкісним видам;

– не спеціалізується виключно на тваринах, а є універсальним інструментом для розпізнавання різних об'єктів.

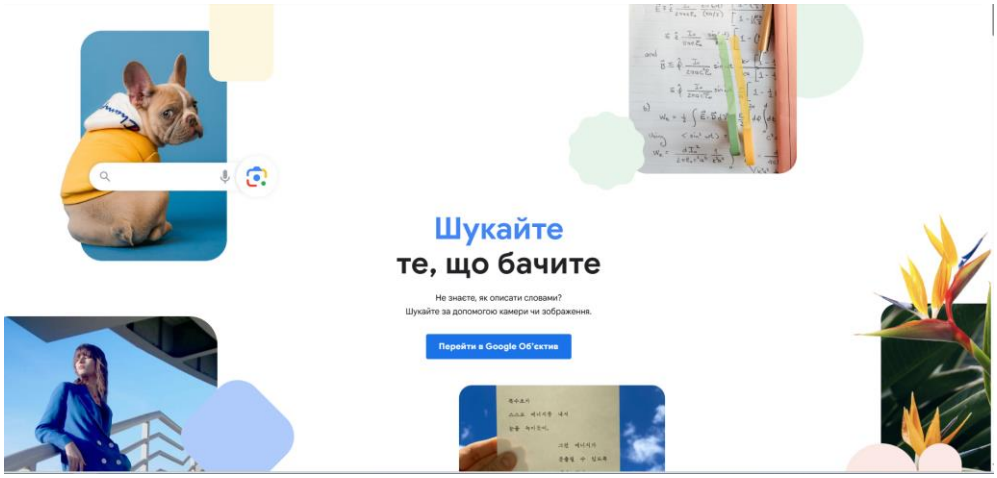

Рисунок 1.1 – сайт Google Lens

Джерело[1]

iNaturalist – це популярна платформа для моніторингу природи, яка дозволяє користувачам завантажувати фотографії тварин і рослин для їх ідентифікації. Платформа використовує штучний інтелект для автоматичного розпізнавання видів, але також дозволяє спільноті експертів доповнювати результати та перевіряти правильність класифікації. iNaturalist має велику базу даних, що включає тисячі видів тварин та рослин по всьому світу.

Переваги iNaturalist:

– Висока точність класифікації завдяки спільній перевірці експертами.

– Багатий набір видів тварин і рослин.

– Доступність для широкої аудиторії користувачів.

Недоліки:

- Процес класифікації може бути затягнутим, оскільки велика частина перевірки здійснюється спільнотою.

- Обмежена кількість інтерактивних функцій для користувачів (наприклад, збереження історії запитів).

- Платформа має більше фокус на наукових дослідженнях, що робить її менш зручною для звичайних користувачів.

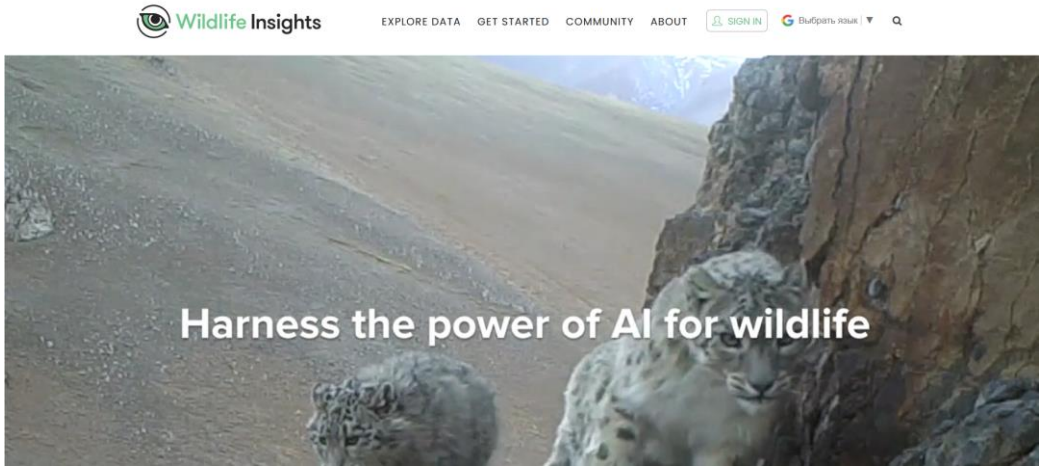

Рисунок 1.2 – сайт Wildlife Insights

Джерело[2]

Wildlife Insights (рис 1.2) – це інноваційна платформа для автоматичної класифікації зображень диких тварин, отриманих через камери спостереження. Цей інструмент використовує передові технології машинного навчання для класифікації зображень, зокрема для ідентифікації тварин у природному середовищі. Платформа розроблена для досліджень біорізноманіття та моніторингу популяцій тварин.

Переваги Wildlife Insights:

– висока точність у класифікації диких тварин завдяки спеціалізованим моделям;

– платформа, орієнтована на збереження біорізноманіття;

– підтримка великої кількості видів тварин.

У табл. 1.1 зображено порівняння вищевказаних сервісів з точки зору задачі розпізнавання зображень тварин.

Існує значний потенціал у застосуванні технологій машинного навчання для класифікації тварин на зображеннях, але існують певні обмеження у функціональності існуючих сервісів. Більшість з них мають широку базу даних і дозволяють ідентифікувати безліч видів, однак не завжди забезпечують точність для специфічних або рідкісних тварин. Більшість платформ не оптимізовані для використання у вигляді інтерактивних чат-ботів, що є важливою конкурентною перевагою розробки Telegram-бота.

Telegram-бот «AIPAWS» має низку конкурентних переваг, які дозволяють йому виділитися серед існуючих аналогів і забезпечити високу ефективність у сфері автоматичної класифікації зображень тварин.

Однією з основних переваг є зручність використання. Завдяки інтеграції з Telegram, користувачам не потрібно встановлювати додаткові програми або проходити складний процес реєстрації. Достатньо надіслати зображення тварини в бот, і він миттєво поверне відповідь із визначеним видом та додатковою інформацією. Це суттєво відрізняє розробку від веб-сервісів і мобільних додатків, які потребують більше часу на запуск та навігацію.

Ще одним важливим аспектом є швидкість обробки зображень. Бот використовує оптимізовані нейронні мережі, які забезпечують високу продуктивність і швидке розпізнавання без значних затримок. Це особливо актуально для користувачів, яким потрібен миттєвий результат, наприклад, для ідентифікації тварин у польових умовах або під час навчального процесу.

Таблиця 1.1 – Порівняння аналогів

Характеристика	Google Lens	iNaturalist	Wildlife Insights
Ціль	Загальне розпізнавання об'єктів	Розпізнавання живих організмів для науки та громадських спостережень	Автоматичне розпізнавання тварин на фото з камер пасток для екологічного моніторингу
Спеціалізація на тваринах	Низька – не спеціалізується виключно на тваринах	Висока – орієнтована на флору та фауну	Дуже висока – спеціалізується лише на диких тваринах
Платформа / Доступ	Мобільні пристрої (Android, iOS), веб	Мобільний додаток, веб-платформа	Веб-платформа (часто використовується дослідниками та установами)
Алгоритм розпізнавання	Приватна Google	Мережа глибокого навчання, натренована на даних спільноти	Глибоке навчання, натреноване на мільйонах фото з пасток
Необхідність підключення	Так	Так	Так
Можливість участі користувача	Ні	Так (користувачі можуть підтверджувати або змінювати ідентифікацію)	Обмежена – в основному для дослідників
Точність у визначенні тварин	Середня для тварин	Висока (залежить від якості фото та виду)	Дуже висока для видів, які вже є в базі
Підтримка рідкісних/локальних видів	Обмежена	Висока (багато локальних видів)	Висока, якщо є відповідні дані у навчальній вибірці
Додаткові функції	Переклад, пошук товарів тощо	Геолокація, журнал спостережень, інтеграція з GBIF	Статистика популяцій, часова активність, екологічний аналіз

Висока точність розпізнавання досягається за рахунок використання сучасних моделей глибокого навчання, зокрема згорткових нейронних

мереж, спеціально навчених на великих наборах даних із зображеннями тварин. Це дозволяє боту коректно класифікувати навіть складні випадки, такі як зображення з нестандартним освітленням або тварини у незвичних позах.

Гнучкість і масштабованість також є ключовими перевагами розробки. Telegram-бот можна легко оновлювати, покращуючи його алгоритми та додаючи нові можливості без необхідності перевстановлення програмного забезпечення користувачами. Крім того, система підтримує обробку великої кількості запитів одночасно, що робить її зручною для широкого використання, зокрема у наукових і навчальних цілях.

Інтеграція з додатковими інформаційними ресурсами підвищує цінність сервісу. Після розпізнавання тварини бот може надавати не лише її назву, але й короткий опис, поширені місця проживання, особливості поведінки та екологічний статус. Це робить його корисним інструментом для студентів, дослідників, натуралістів і всіх, хто цікавиться дикою природою.

Telegram-бот також відзначається високою доступністю. Його можна використовувати на будь-якому пристрої, де є месенджер Telegram, що робить його значно зручнішим у порівнянні з альтернативними рішеннями, які часто залежать від платних підписок, складної інтеграції або необхідності високих технічних вимог.

Таким чином, конкурентні переваги розробки включають зручність використання, швидкість і точність роботи, гнучкість оновлення, інтеграцію з інформаційними базами та широку доступність.

1.3 Постановка задачі

Розробка системи автоматичної класифікації зображень тварин потребує чіткої постановки задачі, яка включає визначення основних вимог до програмного забезпечення, вибір методів машинного навчання, а також забезпечення ефективної інтеграції моделі у зручний для користувача

інтерфейс. Основною метою є створення алгоритму, здатного точно та швидко класифікувати зображення тварин, надаючи користувачам інформативні результати.

Для цього необхідно розробити методику попередньої обробки зображень, яка дозволить покращити якість вхідних даних та зменшити похибки класифікації. Використання згорткових нейронних мереж є оптимальним підходом, оскільки вони демонструють високу ефективність у завданнях розпізнавання образів. Важливо визначити оптимальну архітектуру нейромережі, яка забезпечить баланс між точністю класифікації та швидкістю обробки.

Система повинна мати можливість працювати з широким спектром зображень, враховуючи різні умови освітлення, кути зйомки та варіативність представників одного виду. Для цього необхідно застосувати методи аугментації даних та розробити алгоритми обробки зображень, які допоможуть зробити модель більш стійкою до реальних умов використання.

Окремою задачею є створення програмного інтерфейсу, який забезпечить користувачам зручний спосіб взаємодії із системою. Це включає розробку засобу завантаження зображень, механізмів обробки запитів у реальному часі та надання результатів класифікації у зрозумілому форматі. Оптимізація швидкості роботи моделі та мінімізація затримок при взаємодії з користувачем є критичними факторами для успішного функціонування розробки.

Також необхідно забезпечити масштабованість та адаптивність системи, що дозволить у майбутньому розширювати її можливості, додаючи нові категорії тварин або вдосконалюючи методи класифікації. Це вимагає використання модульного підходу до проєктування програмного забезпечення, що полегшить оновлення моделі та інтеграцію нових технологій.

Висновки до розділу 1

Проведено дослідження предметної області автоматичної класифікації зображень тварин на основі методів комп’ютерного зору та глибокого навчання.

Проаналізовано існуючі програмні рішення, що використовуються для класифікації зображень, визначено їх сильні та слабкі сторони, а також можливості для покращення. Виявлено, що більшість аналогічних продуктів зосереджені на вузькоспеціалізованих категоріях або потребують значних обчислювальних ресурсів, що обмежує їх доступність.

Визначено потенційні конкурентні переваги розробки, що включають використання ефективних моделей машинного навчання, оптимізацію продуктивності та створення зручного користувацького інтерфейсу. Інтеграція сучасних технологій дозволяє досягти високої точності класифікації та швидкої обробки запитів.

Сформульовано основні завдання, які необхідно вирішити в рамках реалізації проєкту. Особлива увага приділяється вибору архітектури нейромережі, підготовці навчальних даних, забезпеченню стійкості системи до варіативності вхідних зображень та розробці механізму взаємодії користувача з системою.

Використання автоматизованих методів аналізу зображень та ефективних алгоритмів обробки даних дозволяє створити інноваційний продукт, здатний значно спростити процес ідентифікації тварин.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТЕЛЕГРАМ-БОТА «AIPAWS» ДЛЯ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ТВАРИН З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

2.1 Проектування структури

Функціональні вимоги. У проєкті буде реалізовано декілька основних функціональних механік, що дозволять користувачу взаємодіяти з телеграм-ботом для отримання передбачень від нейронної мережі. Перш за все потрібно створити саму модель глибокого навчання, яка зможе ідентифікувати зображення, що надсилаються користувачами, та категоріювати їх за категоріями "кішка", "собака", "лисиця", "вовк" і "мавпа". .

В якості основи для побудови моделі було обрано згорткову нейронну мережу (CNN), оскільки саме такі моделі демонструють найкращі результати при роботі з візуальними даними [3]. Основною вимогою було забезпечити здатність моделі навчатися на великій кількості зображень і точно класифікувати їх, зберігаючи при цьому достатню швидкість роботи для інтеграції в режимі реального часу.

Конструкція нейронної мережі передбачала наявність кількох згорткових та підсумовуючих шарів, а також шарів регуляризації для запобігання перенавчанню. Використання функції активації ReLU та фінального шару із сигмоїдною активацією дозволяє ефективно вирішувати задачу бінарної класифікації [4].

Після успішного тренування модель зберігається у форматі .h5, що дає змогу її легко завантажувати без необхідності повторного навчання.

Наступною ключовою функціональною вимогою є розробка телеграм-бота, який забезпечить користувачу простий інтерфейс для взаємодії із нейронною мережею. Основні механіки роботи бота включають прийом зображень, їх передобробку, передачу в модель та відправку результату передбачення назад користувачу.

Телеграм-бот повинен обробляти зображення, змінюючи їх розмір до 150×150 пікселів і нормалізуючи значення пікселів для відповідності вимогам моделі. Також бот має обробляти базові команди /start та /help, які полегшать користування сервісом.

Останньою обов'язковою вимогою є забезпечення стабільного режиму роботи бота через механізм опитування (polling), що дасть змогу працювати без налаштування спеціальної серверної інфраструктури.

Рисунок 2.1 – Діаграма прецедентів

Джерело: розроблено автором

Діаграму прецедентів представлено на рис. 2.1. У запропонованій системі користувач має змогу взаємодіяти з телеграм-ботом шляхом виконання низки функціональних дій. Основними механіками взаємодії є надсилання фото для аналізу, отримання допомоги та отримання відповіді на запит.

При надсиланні фото телеграм-бот ініціює процес завантаження та попередньої обробки зображення, після чого передає його для класифікації за допомогою нейронної мережі, побудованої на базі бібліотеки TensorFlow [5]. На цьому етапі бот забезпечує користувача повідомленнями про поточний статус обробки (наприклад, «Фото отримано», «Триває аналіз» тощо). Після завершення аналізу модель здійснює класифікацію

зображення, визначаючи його належність до однієї з наперед визначених категорій.

Отриманий результат повертається до телеграм-бота, який формує відповідь для користувача у зрозумілому текстовому вигляді. Таким чином, користувач отримує зворотний зв'язок у вигляді повідомлення з результатами обробки надісланого зображення.

Крім основного сценарію надсилання зображення, користувач має змогу скористатися функцією отримання допомоги. За запитом допомоги телеграм-бот надсилає користувачу інструкції щодо правил користування системою, можливостей бота та формату правильного надсилання зображень для аналізу.

Таким чином, взаємодія користувача із системою є інтуїтивно зрозумілою та охоплює усі основні процеси, необхідні для ефективного використання бота за призначенням.

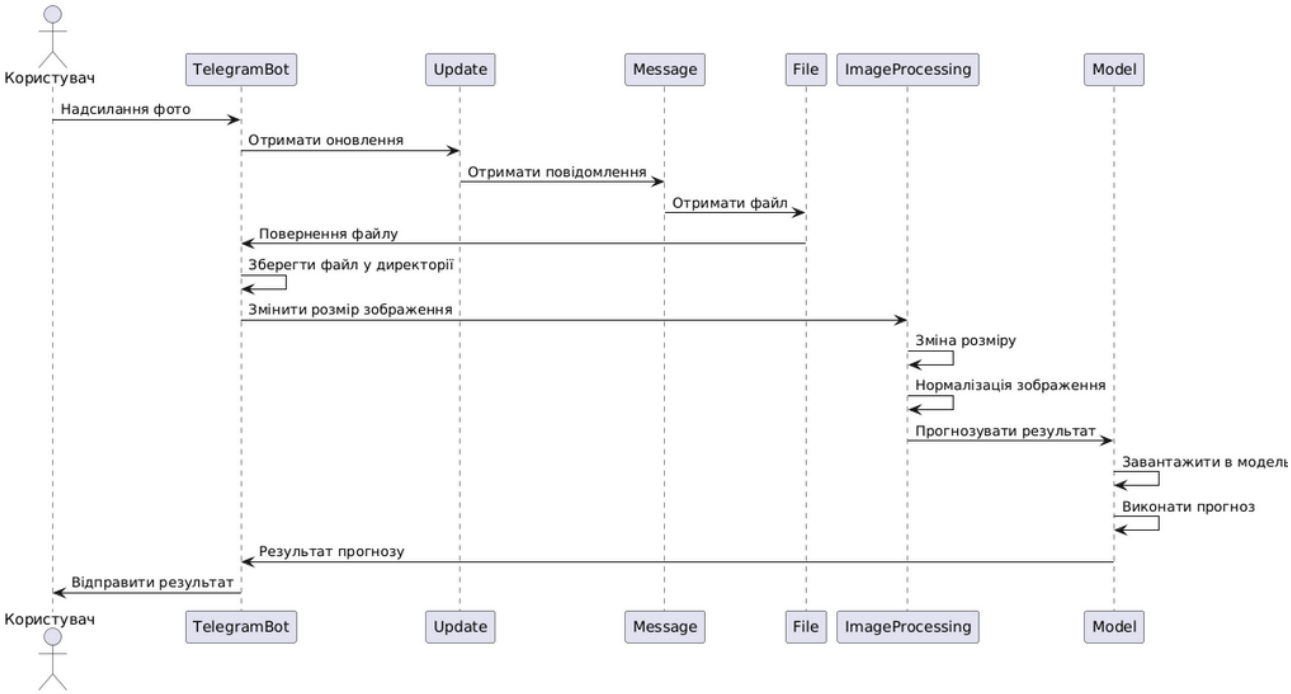

Рисунок 2.2 – Діаграма послідовності

Джерело: розроблено автором

Діаграму послідовності представлено на рис. 2.2. Вона ілюструє об’єкти, які беруть участь у процесі обробки зображення, надісланого користувачем через Telegram-бота, та повідомлення, які передаються між цими об’єктами у процесі виконання одного прецеденту.

Процес починається з того, що користувач надсилає фотографію до Telegram-бота. Бот отримує оновлення (Update) від сервера Telegram, в якому міститься повідомлення (Message) з вкладеним файлом зображення (File). Після отримання повідомлення бот звертається до файлу, отримує його та зберігає у локальну директорію для подальшої обробки.

Далі Telegram-бот ініціює зміну розміру зображення відповідно до вимог моделі машинного навчання. За це відповідає об'єкт ImageProcessing, який виконує зміну розмірів зображення та його нормалізацію (приведення значень пікселів до потрібного діапазону для кращої роботи нейронної мережі).

Після підготовки зображення об'єкт ImageProcessing передає його до об'єкта Model. На цьому етапі модель завантажує отримане зображення у свій вхідний шар та виконує прогноз (класифікацію об’єкта на зображенні). Результат прогнозу повертається назад до об'єкта ImageProcessing, а потім передається Telegram-боту.

Telegram-бот надсилає результат прогнозу користувачу у вигляді текстового повідомлення.

Таким чином, діаграма відображає послідовність усіх кроків, починаючи з надсилання зображення користувачем і закінчуючи отриманням ним результату аналізу. В процесі взаємодіють такі основні об'єкти: TelegramBot, Update, Message, File, ImageProcessing та Model.

2.2 Моделювання даних

На рис. 2.3 представлено діаграму класів. В діаграмі класів розписані класи та атрибути телеграм-бота. Давайте розберемо їх більш детально.

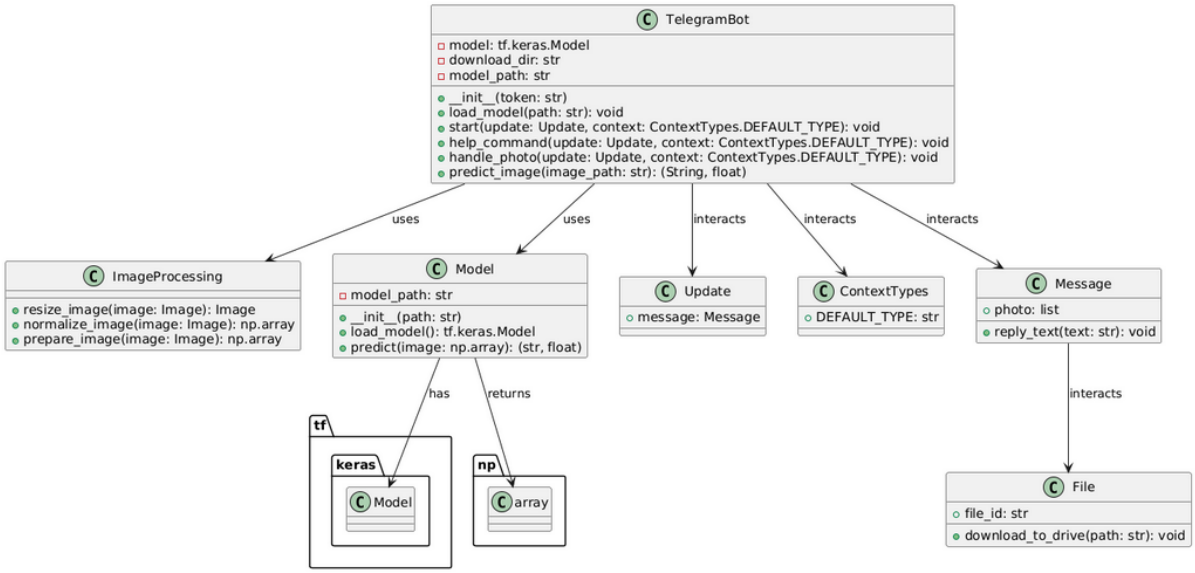

Рисунок 2.3 – Діаграма класів

Джерело: розроблено автором

Клас TelegramBot: Клас, який реалізує Telegram-бота, що взаємодіє з користувачами через Telegram API[6]. Цей бот завантажує модель машинного навчання та використовує її для прогнозування на зображеннях, які надсилають користувачі. Атрибути класу відображені на табл. 2.1, методи – на табл. 2.2.

Таблиця 2.1 – Атрибути класу TelegramBot

Назва	Тип	Опис
model	tf.keras.Model	Змінна, яка зберігає модель машинного навчання, використовуючи TensorFlow[7].
download_dir	str	Шлях до директорії, де бот зберігає завантажені файли.
model_path	str	Шлях до файлу моделі.

Таблиця 2.2 – Методи класу TelegramBot

Назва методу	Повертає	Опис
__init__(token: str)	–	Конструктор, який ініціалізує бота з токеном для підключення до Telegram API.
load_model(path: str)	void	Завантажує модель з вказаного шляху.
start(update: Update, context: ContextTypes.DEFAULT_TYPE)	void	Метод, що ініціює взаємодію з користувачем при отриманні команди "start".
help_command(update: Update, context: ContextTypes.DEFAULT_TYPE)	void	Метод для обробки команди "help", щоб надати допомогу користувачу.
handle_photo(update: Update, context: ContextTypes.DEFAULT_TYPE)	void	Метод для обробки отриманих зображень від користувача.
predict_image(image_path: str)	(String, float)	Метод для прогнозування на зображенні. Повертає результат у вигляді рядка та ймовірності[8].

Клас ImageProcessing: Клас для обробки зображень перед їх передачею моделі машинного навчання. Методи класу відображені на таблиці 2.3.

Клас Model: Клас для роботи з моделлю машинного навчання. Завантажує модель з файлу та виконує передбачення на зображеннях. Атрибути класу відображені на таблиці 2.4, методи – на таблиці 2.5.

Клас Update: Клас, який представляє оновлення від Telegram API, що містить повідомлення, які бот отримує від користувачів. Атрибути класу відображені на таблиці 2.6.

Клас ContextTypes: Клас, який визначає типи контексту, які можуть бути використані для взаємодії з Telegram API. Атрибути класу відображені на таблиці 2.7.

Таблиця 2.3 – Методи класу ImageProcessing

Назва методу	Повертає	Опис
resize_image(image: Image)	Image	Змінює розмір зображення до заданих параметрів перед його передачею до моделі.
Normalize_image(image: Image)	np.array	Нормалізує зображення, перетворюючи пікселі до діапазону від 0 до 1.
Prepare_image(image: Image)	np.array	Готує зображення у формат, сумісний із моделлю: виконує зміну розміру та нормалізацію.

Таблиця 2.4 – Атрибути класу Model

Назва	Тип	Опис
model_path	str	Шлях до файлу, який містить модель машинного навчання.

Таблиця 2.5 – Методи класу Model

Назва методу	Повертає	Опис
__init__(path: str)	–	Конструктор, який ініціалізує шлях до файлу моделі.
load_model()	tf.keras.Model	Завантажує модель машинного навчання з файлу.
predict(image: np.array)	(str, float)	Виконує передбачення на зображенні та повертає результат і ймовірність.

Таблиця 2.6 – Атрибути класу Update

Назва	Тип	Опис
message	Message	Об'єкт, що містить повідомлення користувача.

Таблиця 2.7 – Атрибути класу ContextTypes

Назва	Тип	Опис
DEFAULT_TYPE	str	Тип за замовчуванням для контексту

Клас Message: Клас, який представляє повідомлення, що отримує бот від користувача, включаючи фотографії або текстові повідомлення.

Атрибути представлені на таблиці 2.8

Таблиця 2.8 – Атрибути класу Message

Назва	Тип	Опис
photo	list	Список фотографій, надісланих користувачем.
Reply_text(text: str)	void	Метод для відповіді текстом на повідомлення користувача.

Клас File: Клас, що представляє файл, завантажений з Telegram, включаючи зображення або інші файли.

Атрибути представлені в таблиці 2.9

Таблиця 2.9 – Атрибути класу File

Назва	Тип	Опис
file_id	str	Ідентифікатор файлу.
download_to_drive(path)	void	Метод для завантаження файлу на диск за вказаним шляхом.

Зв'язки між класами:

– TelegramBot використовує клас ImageProcessing для обробки зображень і клас Model для виконання прогнозів;

– TelegramBot взаємодіє з класами Update, ContextTypes, і Message для обробки повідомлень і команд від користувачів;

– клас Message взаємодіє з класом File, що дозволяє отримувати файли, надіслані користувачами;

– клас Model має залежність від tf.keras.Model для виконання передбачень на зображеннях.

У діаграмі діяльності (рис. 2.4) продемонстровано, як Telegram-бот обробляє взаємодію з користувачем.

Рис. 2.4 – Діаграма діяльності

Джерело: розроблено автором

Спочатку бот очікує отримання повідомлення від користувача. Далі перевіряється, чи містить повідомлення фото. Якщо так, бот отримує файл з фотографією, зберігає його у локальну директорію та передає на попередню обробку. Під час обробки відбувається зміна розміру

зображення та його нормалізація для подальшої роботи моделі. Після цього підготовлене зображення передається моделі TensorFlow, яка виконує класифікацію та генерує прогноз. На основі результату класифікації бот формує відповідь і надсилає її користувачеві. Якщо ж користувач надсилає не фото, то бот інформує його про необхідність надіслати зображення. Після виконання будь-якого з сценаріїв бот повертається до початкового стану очікування нового повідомлення.

Висновок до розділу 2

Під час проектування було розроблено всі основні механіки роботи Telegram-бота, такі як: обробка вхідних повідомлень, обробка зображень та передбачення результатів за допомогою нейронної мережі. Також було пропрацьовано UML-діаграми для кращого розуміння структури, логіки взаємодії об'єктів та послідовності процесів у майбутньому програмному продукті. Були створені такі діаграми як: діаграма прецедентів, діаграма послідовності та діаграма діяльності. Всі вони значно допоможуть при реалізації та подальшому удосконаленні Telegram-бота.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТЕЛЕГРАМ-БОТА «AIPAWS» ДЛЯ АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ТВАРИН З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

3.1 Особливості реалізації

Збір даних для моделі. Для навчання нейронної мережі, яка розпізнає зображення тварин, було зібрано набір даних, що охоплює п’ять основних класів: собаки, коти, вовки, лисиці та мавпи. Джерелом зображень стали відкриті датасети, розміщені на платформі Kaggle. Цей ресурс є одним із найбільш популярних серед дослідників та розробників систем штучного інтелекту завдяки великій кількості якісних і структурованих даних.

Зображення у кожному з датасетів були отримані з різних джерел, тому вони відрізнялися за якістю, роздільною здатністю, фоном та умовами освітлення. Після завантаження даних було проведено попередню обробку, необхідну для покращення якості навчання моделі. Було звернено увагу на балансування класів. Оскільки деякі датасети містили значно більше прикладів, ніж інші, їх обсяг було скориговано таким чином, щоб усі класи мали приблизно однакову кількість зображень. Це дозволяє уникнути зміщення в бік частіше представлених категорій [9].

У результаті було сформовано чистий, збалансований та готовий до навчання датасет, який став основою для побудови моделі розпізнавання тварин.

Підготовка даних для обробки. Для ефективного навчання нейронної мережі критично важливо мати якісний і чистий набір даних. На практиці великі датасети часто містять пошкоджені або некоректні файли, які можуть спричинити помилки при завантаженні або негативно вплинути на результати навчання, знижуючи точність класифікації та ускладнюючи оптимізацію моделі.

З цієї причини на першому етапі реалізації було проведено попереднє очищення датасету. Першим кроком стала ітерація по всім категоріям. Програма проходила по всім зображенням, що зберігаються в окремих папках відповідно до класів: Cat, Dog, Wolf, Fox та Monkey. Кожна папка містить фотографії одного типу тварин – котів, собак, вовків, лисиць та мавп.

Далі відбувається перевірка кожного окремого файлу. Для цього файл відкривається у режимі читання байтів (rb), після чого зчитуваються перші кілька байтів вмісту. Основним критерієм перевірки була наявність сигнатури "JFIF" – характерної послідовності байтів, яка свідчить про те, що файл є справжнім JPEG-зображенням. Якщо ця сигнатура була відсутня або під час відкриття виникала будь-яка помилка (наприклад, файл виявлявся пошкодженим, мав неправильну структуру або був неповним), зображення визнавалося некоректним[10].

Усі файли, які не пройшли перевірку, автоматично видалялися з файлової системи. У процесі видалення також вівся підрахунок загальної кількості таких файлів. Для кожного видаленого зображення виводилося інформаційне повідомлення з зазначенням шляху до файлу та причини його видалення. Це дозволяло не тільки контролювати процес, але й забезпечувало прозорість виконання.

На завершення очищення здійснювалося логування усього процесу. Користувачеві поступово виводилися повідомлення про поточний стан перевірки: скільки файлів уже оброблено, з яких саме папок відбувалося читання та скільки зображень було остаточно видалено. Це дало змогу оцінити обсяг проблемних даних і переконатися в тому, що залишився лише якісний матеріал.

Завдяки цьому процесу вдалося значно покращити якість датасету, забезпечити його структурну цілісність і запобігти непередбачуваним збоям під час тренування нейронної мережі, що особливо важливо при роботі з великими наборами зображень у багатокласовій задачі класифікації [11].

Створення генераторів даних. Після очищення датасету наступним важливим кроком була підготовка даних для подачі в нейронну мережу. Для цього використовувався механізм генераторів даних (ImageDataGenerator) із модуля Keras. Цей інструмент дозволяє під час навчання поступово подавати дані пакетами (батчами), а не завантажувати весь набір у пам'ять, що є критично важливим для великих датасетів[12]. Крім того, він автоматично здійснює попередню обробку зображень, зокрема нормалізацію та аугментацію (тобто розширення варіацій даних).

Процес створення генераторів складався з кількох етапів. Першим кроком була нормалізація пікселів: кожен піксель зображення ділився на 255, що призводило до приведення значень до діапазону {0, 1}. Це важливо, оскільки нейронні мережі працюють стабільніше, коли вхідні дані мають невеликі значення {7}.

Наступним етапом стала аугментація даних. Для навчальної вибірки застосовувалися різні методи трансформації: випадкове обертання зображення на кут до 20 градусів (rotation_range=20), масштабування з випадковим збільшенням або зменшенням розміру (zoom_range=0.2), а також горизонтальне віддзеркалення (horizontal_flip=True). Ці операції допомагають моделі не "зазубрювати" фіксовані шаблони з навчальної вибірки, а навчитися розпізнавати об'єкти в різних положеннях і масштабах[13].

Далі дані було розділено на навчальну та валідаційну вибірки. За допомогою параметра validation_split=0.2 автоматично виділили 20% даних для валідації. Таким чином, 80% зображень використовувалося для навчання, а решта 20% – для оцінки якості навчання після кожної епохи.

На завершення було створено два окремих генератори: train_generator для навчальної вибірки (із параметром subset='training') та validation_generator для валідаційної (із параметром subset='validation'). Обидва генератори завантажують зображення безпосередньо з папок, змінюють їхній розмір до 150×150 пікселів (target_size=(150, 150)), подають

дані партіями по 32 зображення (batch_size=32) та використовують режим categorical для визначення цільових міток (кіт, собака, лис, вовк або мавпа).

Таким чином, генератори забезпечили безперервний потік даних із автоматичним попереднім обробленням, що дозволило суттєво оптимізувати процес навчання нейронної мережі [14]

Архітектура нейронної мережі. Після підготовки даних було розроблено архітектуру згорткової нейронної мережі (CNN) для вирішення задачі категоризації зображень.

Структура мережі зображена на таблиці 3.1.

Таблиця 3.1 – Архітектура згорткової нейронної мережі

№	Тип шару	Параметри	Призначення / Коментар
1	Вхідний шар	Розмір зображення: 150×150×3	Вхідне зображення з 3 кольоровими каналами (RGB)
2	Conv2D	32 фільтри 3×3, активація: ReLU	Виявлення базових ознак (краї, лінії)
3	MaxPooling2D	Вікно: 2×2	Зменшення розмірності, збереження найважливіших ознак
4	Conv2D	64 фільтри 3×3, активація: ReLU	Виявлення складніших ознак (напр. Частини тіла тварин)
5	MaxPooling2D	Вікно: 2×2	Як вище
6	Conv2D	128 фільтрів 3×3, активація: ReLU	Розпізнавання абстрактних ознак
7	MaxPooling2D	Вікно: 2×2	Як вище
8	Flatten	–	Перетворення 3D у 1D вектор

Таблиця 3.1 – Архітектура згорткової нейронної мережі

№	Тип шару	Параметри	Призначення / Коментар
9	Dense	512 нейронів, активація: ReLU	Побудова логічних залежностей, обробка ознак
10	Dropout	0.5	Регуляризація, зниження ризику перенавчання
11	Dense (вихідний)	1 нейрон, активація: Softmax	Видача ймовірності класу (1 з 5 класів)

Компіляція моделі. На етапі компіляції нейронна мережа була підготовлена до навчання шляхом визначення основних параметрів, які впливають на процес оновлення ваг, оцінку похибки та загальну якість моделі. Ключовими складовими цього етапу стали вибір оптимізатора, функції втрат і метрики оцінювання.

У якості оптимізатора було обрано Adam (Adaptive Moment Estimation) – сучасний і потужний алгоритм оптимізації, який поєднує в собі переваги двох інших методів: AdaGrad і RMSProp. Особливістю Adam є автоматичне підлаштування швидкості навчання (learning rate) для кожного параметра моделі окремо, що забезпечує стабільніший і швидший процес збіжності. Завдяки цьому модель здатна ефективно навчатися навіть на складних і нестабільних даних, не потребуючи вручну налаштовувати швидкість навчання в кожному конкретному випадку[15].

Для оцінки помилки (відхилення передбачення від правильного значення) було використано функцію втрат binary_crossentropy. Ця функція є стандартом для задач бінарної класифікації, де необхідно визначити, до якого з двох класів належить об’єкт (наприклад, кіт або собака, лис або вовк). Binary_crossentropy порівнює ймовірність передбачення моделі з фактичною міткою класу та видає значення втрати, яке відображає, наскільки сильно модель помиляється при поточних параметрах. Чим менше значення цієї функції, тим краще модель справляється із завданням класифікації.

Як основну метрику якості моделі було обрано accuracy – точність класифікації, тобто відсоток правильних передбачень серед усіх передбачень[16]. Ця метрика дозволяє швидко оцінити загальну ефективність роботи моделі, порівнюючи кількість вірно класифікованих зображень із загальною кількістю прикладів.

Після задання всіх вищезгаданих параметрів модель була скомпільована, тобто переведена у форму, готову до навчання. На цьому етапі вона вже могла приймати навчальні дані, обчислювати втрати, оновлювати ваги на основі обраного оптимізатора і поступово покращувати свої результати відповідно до заданої метрики.

Навчання моделі. Процес навчання (тренування) нейронної мережі проводився з урахуванням ряду технічних параметрів і етапів, що забезпечували ефективність і стабільність результату. Модель тренувалася протягом 10 епох. Одна епоха передбачала повний прохід по всій навчальній вибірці, що дозволяло нейромережі поступово оновлювати свої ваги на основі всіх доступних даних.

Для оптимального використання оперативної пам’яті під час навчання застосовувалася пакетна обробка даних: зображення подавалися у вигляді пакетів розміром 32 елементи. Це не тільки зменшувало навантаження на систему, а й сприяло більш плавному оновленню параметрів моделі[17].

На кожній епосі модель обробляла послідовно всі пакети навчальних даних. Після завершення епохи проводилася перевірка моделі на валідаційній вибірці. Така перевірка дозволяла оцінити здатність моделі узагальнювати знання на нові, раніше не бачені приклади, що є критично важливим у задачах класифікації.

Паралельно з навчанням здійснювався автоматичний збір статистики. Зберігалася історія зміни точності (accuracy) та втрат (loss) як для навчальної, так і для валідаційної вибірки. Ці дані стали основою для подальшого аналізу якості навчання та виявлення можливих проблем, таких як перенавчання або нестабільне зростання метрик [18].

Після завершення всіх 10 епох модель демонструвала стабільну якість роботи без ознак сильного перенавчання. Це свідчило про правильну настройку гіперпараметрів та ефективність структури самої нейромережі.

Для візуального аналізу результатів навчання був побудований графік зміни точності моделі[19]. За допомогою бібліотеки matplotlib створено дві лінії: одна відображала точність на навчальних даних (history.history['accuracy']), інша – точність на валідаційних даних (history.history['val_accuracy']). Зіставлення обох кривих дозволило побачити динаміку навчання: чи є синхронне зростання точності, чи спостерігається відрив, який міг би свідчити про перенавчання або недостатнє узагальнення.

Графік було належно оформлено: підписано осі (вісь X – кількість епох, вісь Y – значення точності), додано легенду та заголовок "Точність моделі" для кращого сприйняття результатів.

Після завершення навчання модель було збережено у файл animals_classifier_model.h5. Цей файл містить усю інформацію про архітектуру моделі, її ваги та параметри оптимізатора. Завдяки цьому модель можна з легкістю завантажити в майбутньому для здійснення прогнозів або подальшого донавчання без необхідності повторного навчання з нуля [20].

Реалізація Telegram-бота для використання моделі. Після завершення етапу навчання нейронної мережі та її збереження у вигляді готової моделі виникла необхідність створити зручний механізм взаємодії з нею для кінцевого користувача. Метою було надати можливість легко класифікувати зображення за допомогою моделі без потреби запускати спеціалізоване програмне забезпечення або писати код. Для цього було обрано формат Telegram-бота, як простий, доступний та популярний спосіб реалізації користувацького інтерфейсу з мінімальними технічними вимогами.

Бот був реалізований таким чином, щоб приймати зображення від користувачів у форматі звичайного медіафайлу, відправленого в чат. Після отримання фотографії бот зберігає її на сервері, проводить попередню обробку зображення (зміна розміру до стандартного, нормалізація значень пікселів, приведення до тензорного формату, який розуміє модель) і передає її на вхід нейронній мережі.

Нейронна мережа, натренована на п’яти категоріях (коти, собаки, вовки, лисиці, мавпи), аналізує зображення та повертає ймовірності належності до кожного з класів[21]. Бот обробляє ці результати, визначає клас із найвищою ймовірністю, формує текстову відповідь та надсилає її користувачу у вигляді повідомлення. У відповіді також зазначається рівень впевненості моделі у своєму передбаченні, що дозволяє користувачу оцінити точність розпізнавання.

Таким чином, Telegram-бот виступає як зручний інтерфейс між людиною та нейронною мережею, дозволяючи легко використовувати інструмент машинного навчання для класифікації тварин на фотографіях у режимі реального часу без складних технічних дій з боку користувача. Бот значно розширює практичну цінність створеної моделі, роблячи її доступною для широкої аудиторії.

Основні особливості реалізації бота. Для реалізації Telegram-бота, що інтегрується з нейронною мережею, потрібно було підключити низку бібліотек, кожна з яких виконує специфічну функцію в проекті: Перелік даних бібліотек (або модулів) вказано у таблиці 3.2.

Імпорт цих бібліотек дозволяє збудувати стабільний, багатофункціональний та масштабований Telegram-бот, який може ефективно взаємодіяти з користувачами та обробляти їхні дані за допомогою нейронної мережі.

Таблиця 3.2 – Бібліотеки, використані при створенні бота

№	Бібліотека	Тип	Основне призначення	Приклади використання
1	os	Стандартна	Взаємодія з файловою системою	os.makedirs, os.path.join
2	logging	Стандартна	Логування подій під час виконання програми	logging.basicConfig(level=logging.INFO)
3	tensorflow	Стороння (ML/AI)	Створення та виконання моделе машинного навчання	tf.keras.models.load_model, обробка numpy-масивів
4	Numpy[22]	Стороння (обчислення)	Робота з багатовимірними масивами та числовими операціями	np.array, img_array / 255.0
5	PIL / Pillow[23]	Стороння (графіка)	Обробка зображен	Image.open, .convert('RGB'), .resize((150, 150))
6	telegram, telegram.ext	Стороння (Telegram API	Створення Telegram-ботів, обробка повідомлень, команд, керування ботом	ApplicationBuilder, app.run_polling(), обробка /start, /help

Визначення констант. Щоб зробити програмний код більш структурованим, зручним для підтримки та масштабування, всі ключові параметри, які часто використовуються протягом роботи скрипта, були винесені у вигляді констант на початок програми. Такий підхід дозволяє централізовано змінювати налаштування, не заглиблюючись у код і не шукаючи відповідні значення в тілі програми. Це суттєво знижує ризик виникнення помилок під час редагування та підвищує читабельність і гнучкість проєкту, що особливо важливо при його подальшій модифікації або розширенні.

Однією з таких констант є MODEL_PATH, значення якої вказує шлях до файлу з попередньо натренованою моделлю. Наприклад, "cat_dog_classifier_model.h5" або, у випадку багатокласової класифікації, "animals_classifier_model.h5". Цей файл було створено після навчання нейронної мережі в середовищі TensorFlow, і він містить усю необхідну

інформацію – архітектуру моделі, її ваги, а також параметри оптимізації. Саме цей файл завантажується при запуску Telegram-бота, щоб здійснювати класифікацію без потреби у повторному навчанні.

Наступна важлива константа – IMG_SIZE, що дорівнює (150, 150). Це розмір вхідних зображень, до якого приводиться будь-яка фотографія, що надсилається користувачем. Фіксований розмір є обов'язковою умовою, оскільки модель під час навчання очікувала саме такий формат вхідних даних. Невідповідність розміру під час передбачення може призвести до помилок або неточних результатів класифікації, тому попереднє масштабування зображень є обов'язковим етапом перед їх подачею до моделі.

Крім того, було визначено константу DOWNLOAD_DIR, яка вказує на назву директорії, де тимчасово зберігаються зображення, отримані від користувачів. У нашому випадку це папка "downloads". Зберігання зображень на диск перед їх обробкою має низку переваг: воно дозволяє більш ефективно керувати ресурсами під час обробки великої кількості запитів, дає можливість вести журнал дій для аналізу або відлагодження, а також спрощує обробку зображень стандартними файловими методами Python [24].

Після надсилання користувачем зображення у чат, бот ініціює процес обробки отриманого медіафайлу. Telegram надсилає фотографії у кількох варіантах роздільної здатності. Для забезпечення максимальної якості аналізу система обирає з-поміж них зображення з найбільшою роздільною здатністю.

Далі бот здійснює збереження отриманого зображення у локальну директорію проєкту. Для цього попередньо перевіряється наявність відповідної директорії (downloads), а за її відсутності вона створюється автоматично.

Після збереження файл передається функції обробки зображень. У межах цієї функції зображення відкривається за допомогою бібліотеки

Pillow (PIL), конвертується у формат RGB та змінює свій розмір відповідно до вхідних вимог нейронної мережі (150×150 пікселів). Після масштабування дані зображення нормалізуються (діленням значень пікселів на 255) та перетворюються на формат тензора для передбачення.

Нормалізоване зображення подається на вхід завантаженій раніше моделі глибокого навчання, яка виконує передбачення. Модель видає ймовірність приналежності об'єкта на зображенні до одного з п’яти класів: "Кішка", "Собака", "Вовк", "Лисиця" або "Мавпа". Після цього результат класифікації разом із відсотком упевненості повертається користувачу у вигляді текстового повідомлення.

Таким чином, процес обробки фотографії включає кілька етапів: отримання, збереження, попередню обробку, подання у модель та відправку результату передбачення користувачу.

Обробка команд

Telegram-бот реалізує також обробку спеціальних текстових команд для взаємодії з користувачем. Основними командами, які підтримуються, є /start та /help.

1. Команда /start. При отриманні даної команди бот відправляє користувачу вітальне повідомлення з короткою інструкцією про можливості системи. Мета цієї команди – встановлення початкової взаємодії та пояснення функціоналу.
2. Команда /help. Команда призначена для надання користувачу довідкової інформації щодо способу використання бота. У відповіді користувач отримує нагадування, що для отримання передбачення потрібно просто надіслати фото однієї із зазначених тварин.

Обробка кожної команди відбувається через відповідні функції-обробники. Після їх виклику бот формує і відправляє відповідь у чат користувача. Таким чином, реалізація обробки команд забезпечує інтуїтивно зрозумілу навігацію та полегшує користування ботом.

Запуск бота. Для запуску Telegram-бота використовується архітектура бібліотеки python-telegram-bot, яка дозволяє зручно налаштовувати обробку подій та команд.

Перед початком роботи бот ініціалізується шляхом створення об'єкта застосунку (ApplicationBuilder) із зазначенням токена доступу до Telegram API. Після цього до бота додаються обробники:

- обробники команд /start та /help,
- обробник вхідних зображень.

Після налаштування обробників бот переходить у режим довготривалого опитування (polling). У цьому режимі він постійно підтримує з'єднання із серверами Telegram і відслідковує надходження нових повідомлень або медіафайлів.

Запуск у режимі опитування є простим та надійним способом реалізації роботи невеликого бота без потреби у складній серверній інфраструктурі (наприклад, без використання вебхуків чи виділеного сервера).

Таким чином, запуск бота завершує процес налаштування системи, після чого користувачі можуть розпочати інтерактивну роботу з ним у реальному часі.

Середовище розробки. Для розробки бота було використано середовище розробки PyCharm. Середовище розробки PyCharm, створене компанією JetBrains, є потужним інструментом для програмістів, які працюють з мовою Python, зокрема у сфері машинного навчання та розробки нейронних мереж. Його архітектура орієнтована на підвищення продуктивності розробника шляхом поєднання зручного інтерфейсу, розумного автодоповнення коду, контекстної підказки, автоматичного імпорту та швидкого переходу між файлами. Завдяки інтеграції з науковими бібліотеками, такими як TensorFlow, Keras, PyTorch та NumPy, PyCharm стає ефективним інструментом для реалізації, тестування та налагодження моделей штучного інтелекту.

Інтеграція з Jupyter Notebook та підтримка інтерактивного середовища виконання дає змогу поєднувати переваги IDE та дослідницького підходу до коду. Крім того, PyCharm пропонує вбудовані засоби для візуалізації даних і підтримку наукових інструментів, що дозволяє аналітику або досліднику штучного інтелекту швидко переходити від ідеї до реалізації. Також важливою перевагою є інтеграція з системами керування версіями та відлагоджувачем, що особливо актуально в командних проєктах і під час експериментальної розробки складних нейромережевих архітектур.

На відміну від більш легких текстових редакторів, PyCharm забезпечує повноцінне середовище для професійного розробника: від написання коду до його тестування, профілювання та розгортання. Це робить його однією з найкращих платформ для роботи над задачами у сфері штучного інтелекту та обробки даних, з акцентом на зручність, масштабованість та інтеграцію з сучасним інструментарієм Python-екосистеми.

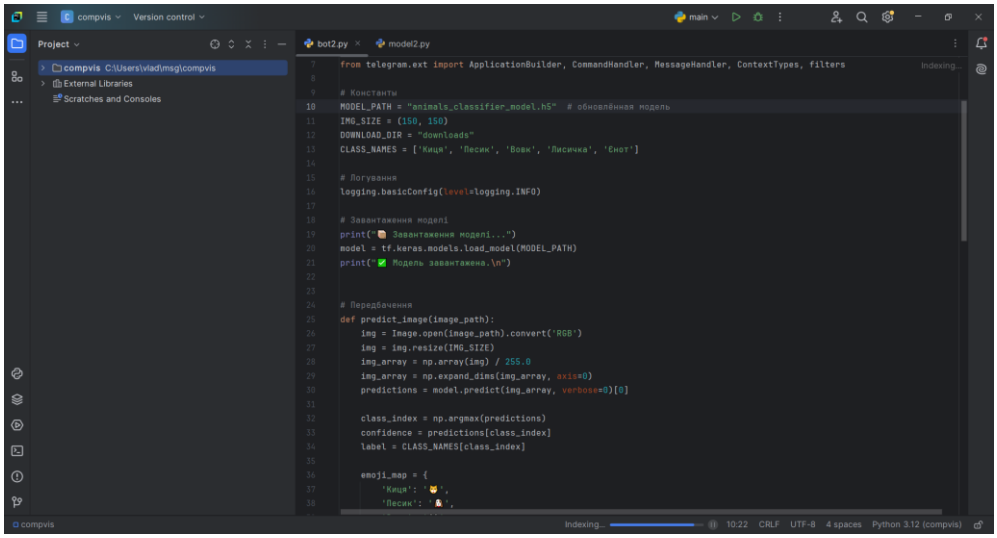

Рисунок 3.1 – Середовище розробки PyCharm

Джерело: розроблено автором

3.2 Тестування

Програмний продукт був протестований за допомогою методів валідації для систем глибокого навчання, перевірений на співпадіння вимог, специфікацій і продуктивності. Головним параметром, на відповідність якому здійснювалась валідація, була точність створеної нейронної мережі. Зроблено це було наступним чином: датасети діляться на дві частини: 80% зображень виділяється для навчання моделі і 20% залишається для подальшого порівняння і перевірки[25]. На кожному етапі навчання відбувалось програмне тестування моделі і оцінка її точності. За 10 ітерацій було досягнуто точності в 83%. На рисунку 3.1 можна побачити сам процес навчання та валідації, відображений у командному рядку. На рисунку 3.3 зображено графік, автоматично згенерований програмою для навчання моделі, на якому відображено зростання точності моделі на кожному етапі і досягнення фінального результату.

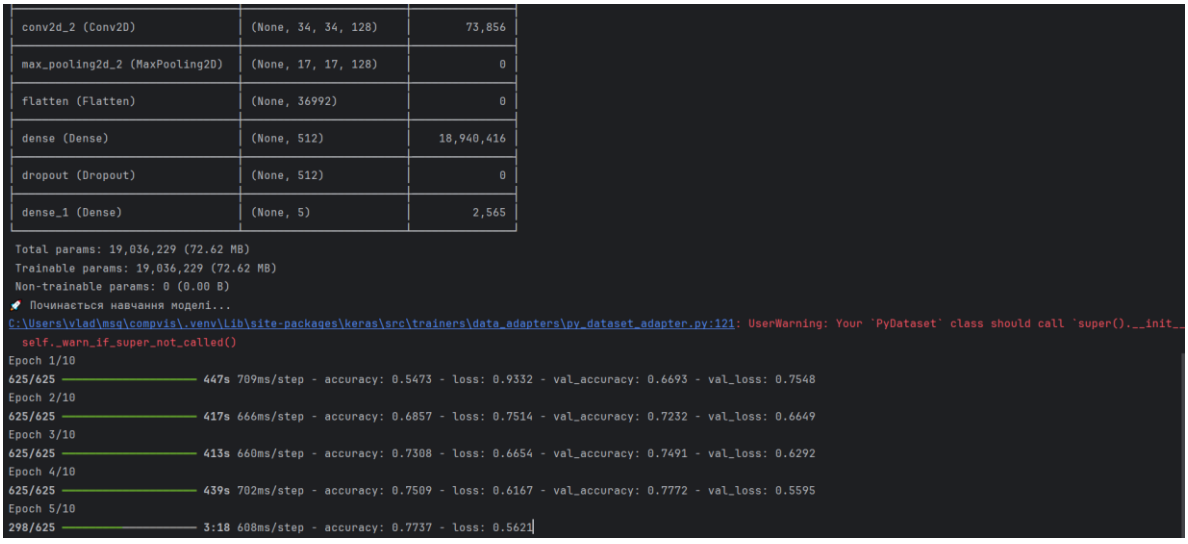

Рисунок 3.2 – Етапи навчання нейронної мережі

Джерело: розроблено автором

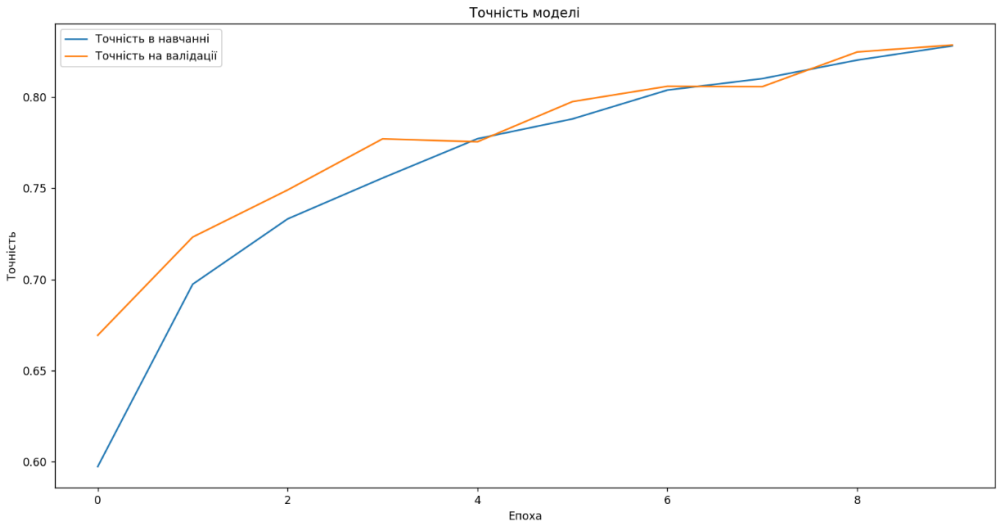

Рис. 3.3 – Графік, що відображає точність моделі на кожній ітерації

Джерело: розроблено автором

3.3 Використання

Для початку роботи з ботом необхідно у додатку Телеграм перейти за тегом @aipaws_test_bot. Після цього користувач побачить стартове вікно бота, яке зображено на рисунку 3.4.

Після цього користувачу необхідно відправити боту команду /start, на що програма запропонує вислати зображення на аналіз. Цей момент зображений на рисунку 3.5.

Тепер користувач може надсилати фото тварин для нейромережевого аналізу. Якщо ж користувач помилково або цілеспрямовано відправить щось окрім файлу зображення, бот обробить цей запит і відповість на нього. Відповідь зображено на рисунку 3.6.

Коли користувач надішле фото чи художнє зображення однієї з тварин, яких модель вміє розпізнавати, бот у відповідь надішле назву тварини на зображенні, відсоток впевненості та у випадку, якщо впевнесність нижче певного відсотка, попросить надіслати ще одне фото для більш точного аналізу. Це зображено на рисунках 3.7 та 3.8.

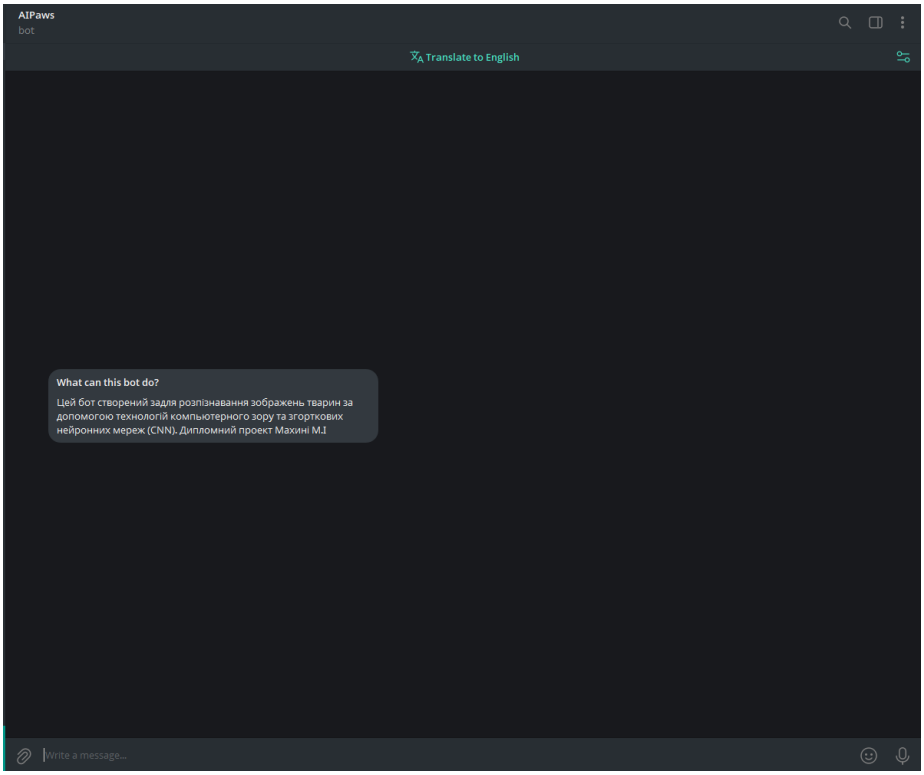

Рисунок 3.4 – Стартове вікно бота AIPaws

Джерело: розроблено автором

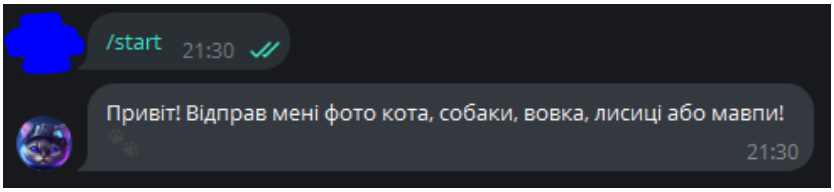

Рисунок 3.5 – Команда /start

Джерело: розроблено автором

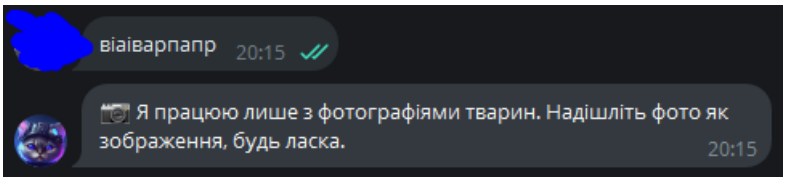

Рисунок 3.6 – Обробка тексту

Джерело: розроблено автором

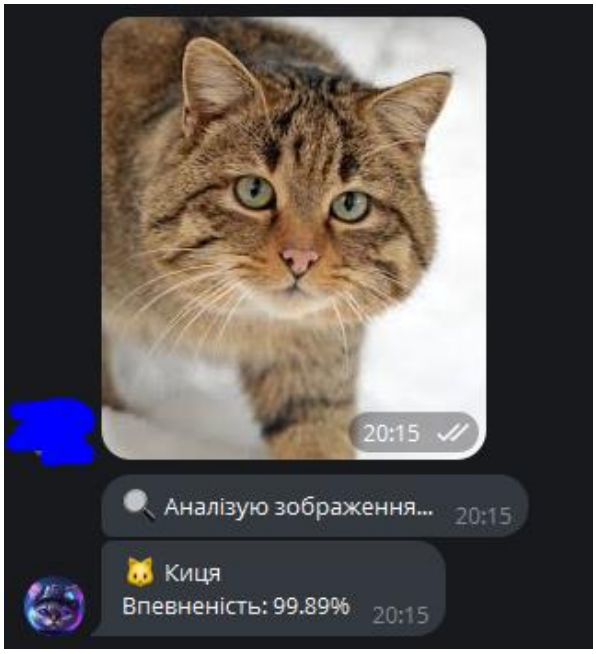

Рисунок 3.7 – Обробка фото тварин

Джерело: розроблено автором

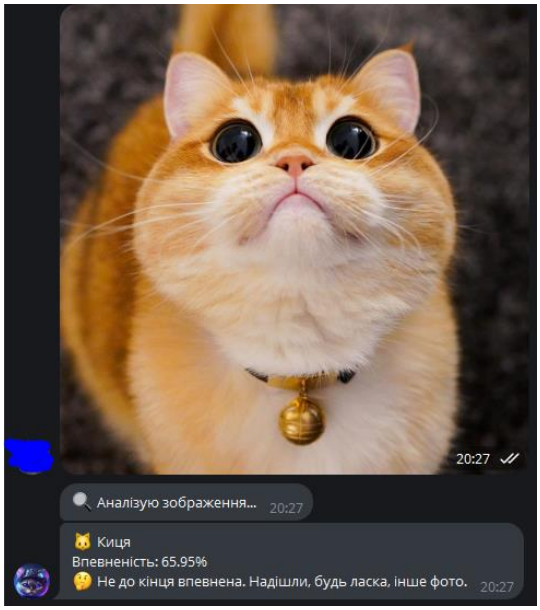

Рисунок 3.8 – Відповідь бота при низькому рівні впевненості

Джерело: розроблено автором

Висновки до розділу 3

Розроблено головну частину програмного продукту, механіки розпізнавання зображень, обробки помилок та взаємодії між ботом і натренованою моделлю. Це надзвичайно важлива частина, тому що є фундаментом всього бота.

Описані всі засоби реалізації, їх особливості, структура та чому саме вони були обрані для створення програмного продукту.

Продукт був протестований для виявлення помилок. Всі знайдені помилки були виправлені, тому в майбутньому вони не стануть складною проблемою.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи були досліджені та виконані поставлені завдання, проаналізовані області застосування програмного продукту та створений опис призначення. Вивчені аналоги та зроблена аналітика по їх перевагам, недолікам та особливостями. Визначені вхідні та вихідні дані, які будуть в готовій моделі, та як вони будуть використані нею.

Визначені функціональні вимоги, які мають бути реалізовані в програмному продукту. Спроєктовано структуру: розроблені діаграми прецедентів, послідовності, класів та діяльності.

Реалізовано програмний продукт. Створено та описано потрібна архітектура, описані класи, методи та атрибути і як вони взаємодіють між собою. Описані засоби реалізації, проаналізовані різні бібліотеки та знайдена та, яка добре підходить під поставлене завдання кваліфікаційної роботи. Протестовано програмний продукт, виявлені та виправлені помилки в деяких механіках.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Google. Google Lens [Електронний ресурс]. URL: https://lens.google/ (дата звернення: 31.03.2025).

2. Wildlife Insights. Global camera trap platform [Електронний ресурс]. URL: https://www.wildlifeinsights.org/ (дата звернення: 31.03.2025).

3. Chollet, F. Deep Learning with Python. New York: Manning Publications, 2017. 361 p.

4. Brownlee, J. Deep Learning for Computer Vision. Australia: Machine Learning Mastery, 2019. 573 p.

5. TensorFlow. TensorFlow official site [Електронний ресурс]. URL: https://www.tensorflow.org/ (дата звернення: 03.05.2025).

6. Telegram. Telegram Bot API [Електронний ресурс]. URL: https://core.telegram.org/bots/api (дата звернення: 03.05.2025).

7. Keras. Keras documentation [Електронний ресурс]. URL: https://keras.io/ (дата звернення: 03.05.2025).

8. Brownlee, J. Machine Learning Mastery [Електронний ресурс]. URL: https://machinelearningmastery.com/ (дата звернення: 03.05.2025).

9. Towards Data Science. Publication on Medium [Електронний ресурс]. URL: https://towardsdatascience.com/ (дата звернення: 03.05.2025).

10. Papers With Code. The latest in machine learning [Електронний ресурс]. URL: https://paperswithcode.com/ (дата звернення: 03.05.2025).

11. Raschka, S., Mirjalili, V. Python Machine Learning. Birmingham: Packt Publishing, 2020. 771 p.

12. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Sebastopol: O'Reilly Media, 2022. 1100 p.

13. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. Cambridge: MIT Press, 2016. 775 p.

14. Patterson, J., Gibson, A. Deep Learning: A Practitioner's Approach. Sebastopol: O'Reilly Media, 2017. 500 p.

15. Smith, L. Machine Learning Systems: Designs that Scale. New York: Manning Publications, 2018. 384 p.

16. Buitinck, L. et al. Guide to scikit-learn [Електронний ресурс]. URL: https://scikit-learn.org/stable/user_guide.html (дата звернення: 03.05.2025).

17. McKinney, W. Python for Data Analysis. Sebastopol: O'Reilly Media, 2022. 544 p.

18. Trotzek, M. Applied Deep Learning with TensorFlow and Keras. Cham: Springer, 2023. 249 p.

19. Turing, J. Neural Networks for Beginners. AI Publishing, 2020. 168p.

20. Троцько, В.В. Методи штучного інтелекту: навчально-методичний і практичний посібник. Харків: ХНУРЕ, 2020. 212 с.

21. Кузнецов, С.В. Машинне навчання і нейронні мережі. Харків: ХНУРЕ, 2020. 148 с.

22. NumPy. Official website [Електронний ресурс]. URL: https://numpy.org/ (дата звернення: 03.05.2025).

23. Pillow. Python Imaging Library documentation [Електронний ресурс]. URL: https://pillow.readthedocs.io/ (дата звернення: 03.05.2025).

24. Коппель, В.Ф. Обробка зображень з використанням Python. Київ: НаУКМА, 2021. 137 с.

25. Хомоненко, О.О. Теорія та практика машинного навчання. Львів: Видавництво ЛНУ, 2020. 231 с.

ДОДАТОК А

```python
import os
import logging
import tensorflow as tf
import numpy as np
from PIL import Image
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, MessageHandler, ContextTypes, filters

# Константи
MODEL_PATH = "animals_classifier_model.h5"  # оновлена модель
IMG_SIZE = (150, 150)
DOWNLOAD_DIR = "downloads"
CLASS_NAMES = ['Киця', 'Песик', 'Вовк', 'Лисичка', 'Єнот']

# Логування
logging.basicConfig(level=logging.INFO)

# Завантаження моделі
print("📦 Завантаження моделі...")
model = tf.keras.models.load_model(MODEL_PATH)
print("✅ Модель завантажена.\n")


# Передбачення
def predict_image(image_path):
    img = Image.open(image_path).convert('RGB')
    img = img.resize(IMG_SIZE)
    img_array = np.array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)
    predictions = model.predict(img_array, verbose=0)[0]

    class_index = np.argmax(predictions)
    confidence = predictions[class_index]
    label = CLASS_NAMES[class_index]

    emoji_map = {
        'Киця': '🐱',
        'Песик': '🐶',
        'Вовк': '🐺',
        'Лисичка': '🦊',
        'Мавпочка': '🐵'
    }
    return f"{emoji_map[label]} {label}", confidence


# Команди
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Привіт! Відправ мені фото кота, собаки, вовка, лисиці або мавпи! 🐾")


async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text(
        "📸 Просто відправ мені фото одного з п’яти тварин: кіт, собака, вовк, лисиця або мавпа.")

```

Рисунок А.1 – Перша частина вихідного коду телеграм-бота

Джерело: розроблено автором

```python
# Обробка фото
async def handle_photo(update: Update, context: ContextTypes.DEFAULT_TYPE):
    photo = update.message.photo[-1]
    file = await photo.get_file()

    os.makedirs(DOWNLOAD_DIR, exist_ok=True)
    file_path = os.path.join(DOWNLOAD_DIR, f"{file.file_id}.jpg")
    await file.download_to_drive(file_path)

    await update.message.reply_text("🔍 Аналізую зображення...")

    label, confidence = predict_image(file_path)
    if (confidence * 100) < 70:
        await update.message.reply_text(
            f"{label}\nВпевненість: {confidence * 100:.2f}%\n🤔 Не до кінця впевнена. Надішли, будь ласка, інше фото.")
    else:
        await update.message.reply_text(f"{label}\nВпевненість: {confidence * 100:.2f}%")


# Обробка інших типів повідомлень
async def handle_text(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("📸 Надішліть фото тварини для розпізнавання (кіт, пес, вовк, лисиця, мавпа).")


async def handle_other(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("📸 Я працюю лише з фотографіями тварин. Надішліть фото як зображення, будь ласка.")


# Запуск
TOKEN = "7565788576:AAHQkeu4K6Ilb7wAV74zyWAD36PrQ1z8eMI"
app = ApplicationBuilder().token(TOKEN).build()

app.add_handler(CommandHandler("start", start))
app.add_handler(CommandHandler("help", help_command))
app.add_handler(MessageHandler(filters.PHOTO, handle_photo))
app.add_handler(MessageHandler(~filters.PHOTO, handle_other))

print("🚀 Бот запущено!")
app.run_polling()
```

Рисунок А.2 – Друга частина вихідного коду телеграм-бота

Джерело: розроблено автором

```python
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Параметри
dataset_path = 'PetImages'
img_height, img_width = 150, 150
batch_size = 32
epochs = 10
class_names = ["Cat", "Dog", "Wolf", "Fox", "Raccoon"]

# Очистка від пошкоджених зображень
def remove_corrupted_images(folder_path, class_names):
    print("🔍 Починаємо перевірку на наявність пошкоджених зображень...")
    num_skipped = 0
    for folder_name in class_names:
        folder_path_full = os.path.join(folder_path, folder_name)
        if not os.path.exists(folder_path_full):
            print(f"❌ Директорія не знайдена: {folder_path_full}")
            continue
        print(f"📂 Обробка директорії: {folder_name}")
        total = len(os.listdir(folder_path_full))
        for idx, fname in enumerate(os.listdir(folder_path_full)):
            fpath = os.path.join(folder_path_full, fname)
            try:
                with open(fpath, "rb") as fobj:
                    is_jfif = tf.compat.as_bytes("JFIF") in fobj.peek(10)
                if not is_jfif:
                    os.remove(fpath)
                    num_skipped += 1
                    print(f"⚠️ Видалено пошкоджений файл: {fpath}")
            except Exception as e:
                os.remove(fpath)
                num_skipped += 1
                print(f"⚠️ Помилка при обробці {fpath}: {e}")
            if (idx + 1) % 1000 == 0:
                print(f"  └─ Перевірено {idx + 1} / {total} файлів у {folder_name}")
    print(f"✅ Перевірка завершена. Видалено {num_skipped} пошкоджених зображень.\n")

remove_corrupted_images(dataset_path, class_names)

# Створення генераторів
print("⚙️ Створення генераторів зображень...")
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True,
)

train_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',  # Для багатокласової класифікації
    subset='training',
    shuffle=True
)
```

Рисунок А.3 – Перша частина вихідного коду для навчання моделі

Джерело: розроблено автором

```python
validation_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    shuffle=False
)

# Створення моделі
print("🧠 Створення моделі...")
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(class_names), activation='softmax')  # 5 класів
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

# Навчання моделі
print("🚀 Починається навчання моделі...")
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator
)
print("✅ Навчання завершено.\n")

# Побудова графіка точності
print("📊 Відображення графіку точності...")
plt.plot(history.history['accuracy'], label='Точність в навчанні')
plt.plot(history.history['val_accuracy'], label='Точність на валідації')
plt.xlabel('Епоха')
plt.ylabel('Точність')
plt.legend()
plt.title('Точність моделі')
plt.show()

# Збереження моделі
print("💾 Збереження моделі у файл animals_classifier_model.h5 ...")
model.save("animals_classifier_model.h5")
print("✅ Модель збережена.")
```

Рисунок А.4 – Друга частина вихідного коду для навчання моделі

Джерело: розроблено автором