МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАТИКИ ТА КОМП'ЮТЕРНОЇ ТЕХНІКИ

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	Інформаційні системи та технології
Освітня програма	Інформаційні системи та технології
Група	6.04.126.010.19.1

ДИПЛОМНИЙ ПРОЄКТ

на тему: «Розроблення застосунку для класифікації зображень із використанням візуального трансформера»

Виконав: студент Антон ЛОБУШКО

Керівник: к.т.н, доц. Олексій ГОРОХОВАТСЬКИЙ

Консультант:

Рецензент: к.т.н, доц. кафедри Інформатики
Харківського національного університету радіоелектроніки
доц. Валентин ЛЮБЧЕНКО

Харків – 2023 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет Інформаційних технологій
Кафедра Інформатики та комп'ютерної техніки
Освітній ступінь Бакалавр
Спеціальність 126 "Інформаційні системи та технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри
Інформатики та комп'ютерної техніки
Проф. Сергій УДОВЕНКО
"01" лютого 2023 р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Лобушко Антону Володимировичу

1. Тема проєкту: «Розроблення застосунку для класифікації зображень із використанням візуального трансформера»

керівник проєкту: Доцент кафедри ІКТ, к.т.н, доц. Гороховатський Олексій Володимирович

затверджені наказом ректора від "01 лютого" 2023 р. № 107-С

2. Строк подання студентом проєкту: 01 червня 2023 року

3. Вихідні дані до проєкту: опис методів класифікації зображень із використанням нейронних мереж, попередньо навчені моделі конволюційних нейронних мереж (CNN), попередньо навчені моделі візуального трансформеру (ViT), бібліотека комп'ютерного зору OpenCV, середовище програмування мовою Python та відповідні бібліотеки для роботи зі штучними нейронними мережами, матеріали проходження переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Дослідження проблеми та постановка задачі.

Розділ 2. Математичні основи візуального трансформера.

Розділ 3. Розробка додатку для класифікації зображень з використанням візуального трансформера.

Розділ 4. Експериментальні дослідження та аналіз результатів.

5. Перелік графічного матеріалу: актуальність проблеми (1 слайд), мета та завдання проєктування (1 слайд), опис моделі візуального трансформера (2-3 слайди), дослідження точності та ефективності реалізованої моделі (2-3 слайди), розроблення застосунку (3-4 слайди), результати порівняння із конволюційними мережами (1 слайд), висновки (1 слайд).

6. Консультація розділів дипломного проєкту. Консультант відсутній.

7. Дата видачі завдання: "01" лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Розроблення плану дипломного проєкту, ознайомлення з літературними джерелами за темою.	01.02.2023 – 28.02.2023	
2.	Аналіз предметної області.	01.02.2023 – 28.02.2023	
3.	Дослідження трансформерів як методу класифікації зображень.	01.03.2023 – 01.04.2023	
4.	Розроблення застосунку для класифікації зображень.	01.03.2023 – 20.05.2023	
5.	Перевірка чернетки дипломного проєкту та внесення змін до неї керівником.	01.04.2023 – 10.04.2023	
6.	Оформлення дипломного проєкту, перевірка якості дипломного проєкту на плагіат.	01.05.2023 - 31.05.2023	
7.	Подання Голові Екзаменаційної комісії щодо захисту дипломного проєкту.	01.06.2023	

Студент Антон Лобушко

Керівник проєкту Олексій Гороховатський

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту: 60 сторінок, 25 рисунків, 8 таблиць, 1 додаток, 28 джерел.

Об'єктом дослідження є технологія обробки зображень із використанням візуального трансформера (Visual Transformer).

Метою дипломного проєкту є розробка застосунку для класифікації зображень з використанням візуального трансформера.

У роботі використовувалися методи аналізу та обробки зображень, а також методи машинного навчання. Розглянуто різні методи класифікації зображень, зокрема й архітектура Visual Transformer. Розроблено програму для класифікації зображень, що використовує архітектуру Visual Transformer. Проведено тестування програми на вибірці даних для оцінки її ефективності.

У дослідженні показано, що застосування візуального трансформера у додатку для класифікації зображень дозволяє досягти високої точності та ефективності при обробці зображень. Отримані результати демонструють потенціал застосування візуального трансформера у різних галузях, включаючи розпізнавання образів, обробку природної мови та аналіз даних.

Ключові слова: візуальний трансформер (Visual Transformer), класифікація зображень, машинне навчання, застосунок, обробка зображень.

ABSTRACT

Explanatory note to the diploma project contains: 60 pages, 25 figures, 8 tables, 1 appendix, 28 spreadsheets.

The object of research is image processing technology using a visual transformer (Visual Transformer).

The goal of the diploma project is to develop an image classification application using a visual transformer.

The work used image analysis and processing methods, as well as machine learning methods. Various methods of image classification are considered, including the Visual Transformer architecture. An image classification program using the Visual Transformer architecture has been developed. The program was tested on a sample of data to assess its effectiveness.

The study shows that the use of a visual transformer in an image classification application allows for high accuracy and efficiency in image processing. The obtained results demonstrate the potential of the application of the visual transformer in various fields, including pattern recognition, natural language processing and data analysis.

Keywords: visual transformer (Visual Transformer), image classification, machine learning, application, image processing.

ЗМІСТ

ВСТУП

В останні роки завдання класифікації зображень стають все більш актуальними в багатьох галузях, таких як охорона здоров'я, автомобільна промисловість та роздрібна торгівля. Однак існуючі методи класифікації не завжди дозволяють досягти високої точності, особливо при обробці великих обсягів даних.

Візуальний трансформер – це сучасний метод класифікації зображень, який дозволяє досягти високої точності при обробці великих обсягів даних. У цьому проєкті проаналізовано застосування технології Visual Transformer для класифікації зображень та розроблено застосунок на основі цього методу.

Метою роботи є розробка застосунку для класифікації зображень на основі візуального трансформера та порівняння його з існуючими методами. Для досягнення поставленої мети були було поставлено такі задачі: дослідження теоретичних основ візуального трансформера; аналіз застосування цього методу для класифікації зображень; розробка додатку на основі візуального трансформера; проведення експериментів для порівняння ефективності розробленого додатку з існуючими методами класифікації.

Результати цієї роботи можуть бути корисні для дослідників і розробників, що працюють в області класифікації зображень, а також бути використані як відправна точка для подальших досліджень в цій області. Крім того, розроблений застосунок може бути використаний в різних галузях, де потрібна класифікація зображень.

1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний опис і аналіз предметної області

Розвиток сучасних технологій збільшив кількість доступних зображень і спричинив появу різноманітних сервісів, заснованих на аналізі зображень. У цьому проєкті проводиться аналіз та огляд предметної області для розробки застосунку для класифікації зображень з використанням візуального трансформера (Visual Transformer).

Модель візуального трансформера була вперше представлена в науковій статті під назвою "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale" [1] («Зображення варте 16x16 слів: трансформери для розпізнавання зображень у великому масштабі»), авторами якої є Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, та Thomas Unterthiner. Ця стаття була опублікована в електронному репозитарії arXiv у жовтні 2020 року.

У цій статті автори представили новий підхід до обробки зображень, використовуючи архітектуру трансформерів, які раніше були популярні у галузі обробки текстів. Вони показали, що візуальні трансформери можуть бути ефективними для класифікації зображень, навіть перевершуючи традиційні відомі підходи, такі як згорткові нейронні мережі (Convolutional Neural Network, CNN). Класифікацією зображень називають завдання машинного навчання, що полягає у визначенні, якому класу (категорії) належить це зображення.

Класифікація зображень за допомогою візуального трансформера є актуальною проблемою в галузі комп'ютерного зору та машинного навчання. Вона може бути використана в багатьох сферах, таких як медицина, робототехніка, автономні транспортні засоби та системи безпеки. Для вирішення цієї задачі використовуються різні методи та алгоритми, в тому числі згорткові нейронні мережі, які є найбільш поширеним методом.

При розробці застосунку для класифікації зображень необхідно враховувати ряд важливих факторів. По-перше, потрібно визначити завдання класифікації і спосіб подання даних. Також важливо вибрати оптимальний набір інструментів та технологій для реалізації проєкту. Одним із ключових інструментів для аналізу предметної області є використання графічного матеріалу, такого як діаграми, графіки та схеми.

При розробці застосунку для класифікації зображень з використанням візуального трансформера слід також враховувати такі фактори, як обсяг даних, точність класифікації і швидкість роботи. При виборі алгоритмів і моделей слід враховувати їхню ефективність і можливості оптимізації.

Для аналізу предметної галузі слід використовувати різні джерела інформації, такі як наукові роботи, публікації, документацію та відгуки про промислові рішення в цій галузі.

Наступним кроком є визначення основних функціональних вимог до застосунку, що розробляється. Для цього необхідно проаналізувати завдання, які повинен вирішувати додаток, і визначити, які функції необхідні для вирішення цих завдань. Наприклад, для завдання класифікації зображень може знадобитися наступний функціонал: завантаження зображень в систему, обробка зображень для знаходження ознак, застосування алгоритму класифікації до отриманих ознак, виведення результатів класифікації.

Визначення вимог до інтерфейсу програми – ще один важливий етап проєктування. Інтерфейс повинен бути зручним та інтуїтивно зрозумілим для користувачів. Необхідно визначити, які елементи інтерфейсу будуть використовуватися для завантаження та відображення зображень, виведення результатів класифікації.

На етапі проєктування також слід визначити технічні вимоги до застосунку. Сюди входить вибір мови програмування та бібліотек, які будуть використовуватися для реалізації програми. Необхідно також визначити вимоги до апаратного забезпечення, на якому працюватиме програма.

Після того, як вимоги до програми визначені, можна приступати до її реалізації. Розробка програми може здійснюватися поетапно, починаючи зі створення прототипу і закінчуючи тестуванням та оптимізацією.

Важливим етапом розробки програми для класифікації зображень є тестування на різних наборах даних. Тестування слід проводити як на наборах даних, що використовувалися під час розробки, так і на нових наборах даних, які не використовувалися під час розробки. Це допоможе перевірити узагальнюючу здатність моделі та її роботу на різних типах зображень.

Для тестування можна використовувати як відкриті набори даних, такі як CIFAR-10, ImageNet та інші, так і власні набори даних, створені під конкретні завдання. При виборі набору даних необхідно враховувати його репрезентативність та різноманітність, щоб тестування було максимально корисним та інформативним.

Одним із важливих параметрів, який слід оцінювати під час тестування, є точність класифікації. Її можна виміряти за допомогою різних метрик, таких як accuracy, precision, recall, F1-score та інших. Крім того, для оцінки якості класифікації можуть використовуватися матриці помилок, криві навчання та криві ROC (Receiver Operating Characteristic curve).

Після тестування необхідно проаналізувати отримані результати, щоб виявити можливі проблеми та помилки. Якщо точність класифікації не відповідає вимогам, то необхідно проаналізувати причини цього і вжити заходи для поліпшення якості моделі. Це можна зробити шляхом зміни гіперпараметрів моделі, використовувати інші алгоритми класифікації або збільшити розмір навчальної вибірки.

В цілому, тестування є важливим етапом у розробці програми для класифікації зображень. Воно дозволяє оцінити якість моделі та її роботу на різних типах зображень, а також виявити можливі проблеми та помилки, які необхідно виправити.

В ході досліджень в цій предметній області буде розглянуто ряд робіт і наукових статей та проаналізовано предметну область розробки програми для класифікації зображень з використанням візуального трансформера,

проаналізовано існуючі додатки для класифікації зображень, а також проводиться аналіз переваг та недоліків даного підходу порівняно з іншими методами класифікації зображень.

1.2 Огляд та аналіз існуючих додатків для класифікації зображень

Сфера штучного інтелекту наразі є однією з провідних галузей в IT, яка дуже швидко розвивається та, за прогнозами, її розвиток буде набирати оберти найближчими роками. Особливої уваги потребує саме графічна інформація, її аналіз, обробка, редагування. Популярність та розповсюдженість інформації у вигляді фото та відео весь час зростає, збільшується кількість галузей, де така інформація застосовується.

До того ж саме фотографії, малюнки, відеозаписи є носіями великої кількості інформації, порівняно із текстом, звуком, тощо.

На ринку штучного інтелекту прогнозується значний ріст у найближчі роки. Згідно з дослідженням MarketsandMarkets, ринок штучного інтелекту зросте з 87 млрд доларів США в 2022 році до 407 млрд доларів США в 2027 році [17]. Це зростання відбувається внаслідок збільшення застосування штучного інтелекту в багатьох галузях, таких як автопілотування, медицина, виробництво, безпека та інші.

Наразі майже всі сфери нашого життя пов’язані з графічними зображеннями – наприклад, камери спостереження на дорогах, на території приватних домоволодінь та підприємств, у громадських місцях, лікарнях, школах, садочках. Сфера медицини також широко використовує саме фотоматеріали та відео – наприклад, знімки УЗД або МРТ, рентгенівські знімки, фотофіксацію проявів захворювань.

Із цього можна зробити висновок, що такий застосунок, як класифікатор зображень, буде затребуваним багатьма людьми, і неодмінно стане корисним.

На ринку вже існує багато застосунків із класифікації зображень у різних галузях. До одного з найпопулярніших можна віднести Google Lens [18], який може, використовуючи зображення, зроблені на камеру телефона, ідентифікувати предмети або живі істоти на фото, а також шукати їх в Інтернеті, на маркетплейсах, в онлайн-енциклопедіях тощо. На картинці нижче (рис. 1.1) наведено приклад роботи цього застосунку.

Рисунок 1.1 – Демонстрація роботи застосунку Google Lens

Також достатньо відомим є застосунок для класифікації зображень від соціальної мережі iNaturalist [19] яке має назву Seek і сфокусоване воно саме на розпізнаванні видів тварини, птахів та комах на основі їх фотографій (рис. 1.2).

Окрім швидко працюючої нейронної мережі, цей застосунок виділяється також геймифікацією своєї роботи – наприклад, воно не лише класифікує тварин за їх фотографіями, але й створює для користувача власну бібліотеку (або каталог) видів тварин, яких він вже бачив та фотографував. Таким чином можна збирати власну колекцію, змагатися з іншими користувачами. Із неочевидних плюсів цього

застосунку – воно спонукає людей проводити більше часу на вулиці, що корисно для здоров'я, звертати увагу на оточуючий світ.

Що стосується використання класифікаторів зображень у медицині – це одне з найбільш важливих напрямків розвитку штучного інтелекту, адже це не тільки зменшить навантаження на лікарів, прискорить їх роботу, але й зробить медичну допомогу більш доступною для людей з бідних регіонів або з регіонів, де лікарів не вистачає на вистачає на всіх людей, які потребують допомоги.

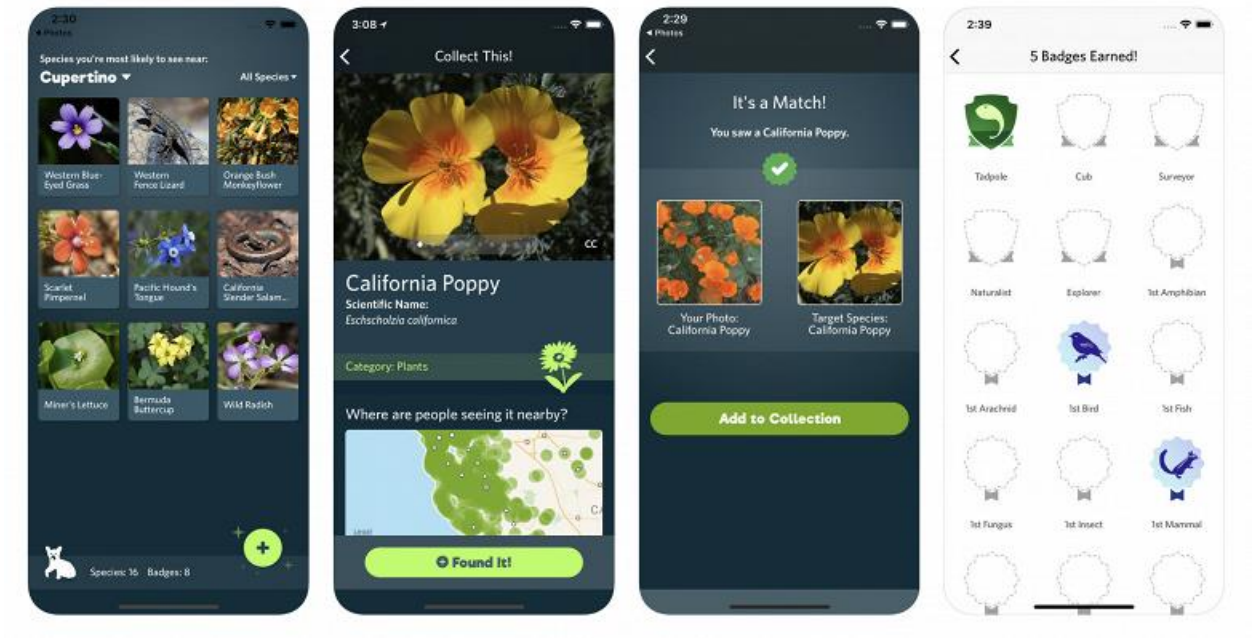


Рисунок 1.2 – Демонстрація роботи застосунку Seek

Гарним представником застосунків-класифікаторів зображень у медичній сфері є сервіс DeepGestalt [20] (рис. 1.3), який може розпізнавати рідкісні генетичні захворювання з вражаючою точністю – більше, ніж 91% правильності.

В цілому діагностика генетичних захворювань є дуже складною темою навіть для фахівців. Проте, генетичні тести зазвичай є не дуже доступними для населення через значну вартість. А для використання DeepGestalt навіть не треба якоїсь дорогої апаратури – лише камера телефону. Наразі цей застосунок може розпізнати близько 200 захворювань.

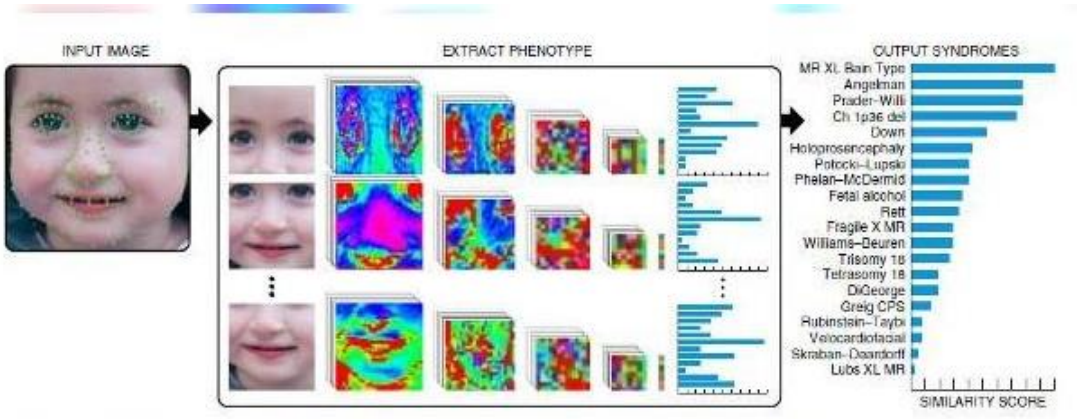


Рисунок 1.3 – Демонстрація роботи застосунку DeepGestalt

Також розпізнавання зображень може суттєво полегшити та покращити роботу правоохоронним органам та органам юстиції – наприклад, розпізнавати номерні знаки та марки машин, що перебільшили допустиму швидкість. Такі системи вже є на дорогах України.

Проте такі рішення є і не лише державними, але й доступними для приватних осіб, наприклад, для власників бізнесу. Одним з прикладів таких застосунків є NumberOK [21] (рис. 1.4).

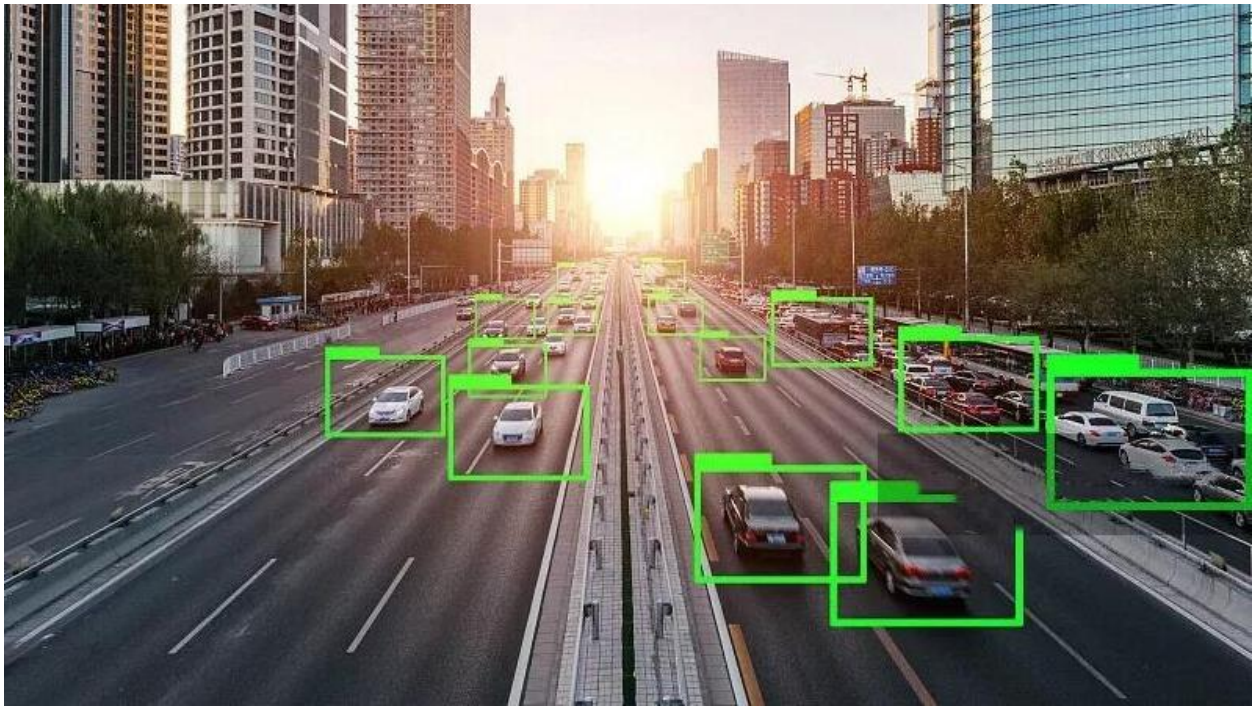


Рисунок 1.4 – Демонстрація роботи застосунку NumberOK

Цей застосунок, розроблений в Чехії, вміє розпізнавати номерні знаки, марку, модель, тип та колір автомобілю, а також напрямок його руху, а також може керувати рухом шлагбаумів або автоматичних воріт, дозволяючи лише певним машинам проїжджати без втручання людини.

1.3 Огляд та аналіз існуючих розв'язків задачі класифікації зображень

Завдання класифікації зображень є одним із найпоширеніших завдань у галузі комп'ютерного зору та машинного навчання. В даний час існує безліч підходів до вирішення цього завдання. Розглянемо деякі з них.

Одним із найбільш популярних підходів є використання згорткових (конволюційних) нейронних мереж (CNN). Згорткові нейронні мережі показують високу точність у задачах класифікації зображень завдяки здатності автоматично витягувати ознаки з зображень на основі згортки з ядрами різних розмірів і форм. Однак, згорткові нейронні мережі мають деякі недоліки, такі як висока кількість параметрів та складність навчання.

Іншим підходом є використання методів машинного навчання на основі глибоких дерев рішень, таких як Random Forest [12] або Gradient Boosting Decision Trees (GBDT) [13]. Ці методи можуть показувати високу точність у завданнях класифікації зображень, але часто вимагають багато часу на навчання і споживають багато обчислювальних ресурсів.

Також існують методи класифікації зображень на основі глибокого навчання, які називають Transformer. Ці методи використовують візуальні трансформери для обробки зображень та показують високу точність у задачах класифікації. Transformer використовує атеншн-механізм (attention) для роботи із зображеннями та дозволяє автоматично виділяти найбільш значущі ознаки. Однак, методи класифікації на основі трансформерів є відносно новими і вимагають великої кількості обчислювальних ресурсів.

Наприклад у статті «Visual Transformers: Token-based Image Representation and Processing for Computer Vision» [2] досліджуєтеся новий підхід до класифікації зображень, використовуючи трансформери для подання та обробки зображень. Вона пропонує представлення зображень у вигляді послідовності токенів та використання архітектури трансформера для аналізу цих токенів. Результати експериментальних досліджень, проведених у статті, свідчать про переваги цього підходу порівняно з традиційними методами обробки зображень. Це відкриває нові перспективи для розвитку методів класифікації зображень.

Існують також методи класифікації на основі генеративних моделей, таких як Generative Adversarial Networks (GAN) [14]. Ці моделі можуть створювати нові зображення на основі наявних і мають здатність до розпізнавання та класифікації зображень. Однак методи на основі GAN мають складну архітектуру і також часто вимагають великої кількості обчислювальних ресурсів.

Таким чином, існує безліч підходів до розв'язання задач класифікації зображень. Деякі з них ґрунтуються на використанні класичних методів машинного навчання, таких як метод опорних векторів (Support Vector Machines, SVM), випадковий ліс (Random Forest), нейронні мережі (Neural Networks) тощо. Інші підходи засновані на використанні глибокого навчання та згорткових нейронних мереж, таких як AlexNet, VGG, GoogleNet, ResNet та ін.

Серед існуючих розв'язків задачі класифікації зображень можна виділити такі найвідоміші архітектури:

1. AlexNet [5] – згорткова нейронна мережа, розроблена у 2012 році для класифікації зображень. AlexNet є однією з перших згорткових нейронних мереж, яка успішно застосовувалася для вирішення задачі класифікації зображень на наборі даних ImageNet.

2. VGG [6] – згорткова нейронна мережа, розроблена у 2014 році, яка була представлена в рамках конкурсу ImageNet Large Scale Visual Recognition Challenge (ILSVRC). VGG має дуже глибоку архітектуру та складається з 16 або 19 шарів.

3. GoogLeNet [7] – згортка нейронна мережа, розроблена в 2014 році командою Google. Вона має дуже глибоку архітектуру і є однією з перших нейронних мереж, в якій використали блок інцептіон (Inception Module).

4. ResNet [8] – згорткова нейронна мережа, розроблена у 2015 році командою Microsoft Research Asia. ResNet має глибоку архітектуру і складається з блоків, які містять пропуск з'єднань (skip connections), що дозволяє боротися з проблемою градієнтів, що загасають, при навчанні.

5. EfficientNet [9] – згорткова нейронна мережа, розроблена в 2019 році, яка є сімейством архітектур, оптимізованих за кількістю параметрів, кількістю операцій і точності на тестовій вибірці ImageNet.

Однак, незважаючи на значний успіх згорткових нейронних мереж у вирішенні задачі класифікації зображень, вони все ще стикаються з певними проблемами, наприклад, їх здатністю до узагальнення. У зв'язку з цим виникає потреба у нових моделях, які можуть справлятися з складнішими завданнями та мають більш високу здатність до узагальнення.

Одним із таких нових підходів є використання візуального трансформера (Visual Transformer). Visual Transformer - це нейронна мережа, яка використовує механізм уваги обробки вхідних зображень. Візуальний трансформер складається з блоків, кожен з яких включає безліч багаторівневих самоуважних (self-attention) механізмів, які можуть шукати взаємодії між різними частинами зображення.

Візуальний трансформер демонструє високу точність у завданнях класифікації зображень і має більш високу здатність до узагальнення порівняно з згортковими нейронними мережами. Також він може використовуватися для інших завдань комп'ютерного зору, наприклад для обробки природних зображень, для генерації зображень і для сегментації зображень.

У цьому дипломному проєкті буде проведено аналіз та розробку програми для класифікації зображень з використанням візуального трансформера.

1.4 Формулювання задачі

Метою є розробка застосунку для класифікації зображень з використанням візуального трансформера (Visual Transformer). Основне завдання проєкту полягає у створенні моделі глибокого навчання, яка зможе класифікувати зображення на основі їхнього вмісту.

Для досягнення цієї мети було поставлено такі завдання:

1. Провести огляд та аналіз предметної галузі задачі класифікації зображень.
2. Вивчити та проаналізувати існуючі методи класифікації зображень на основі згорткових нейронних мереж та візуальних трансформерів.
3. Збір та підготовка даних для навчання моделі.
4. Розробити архітектуру програми для класифікації зображень, засновану на візуальному трансформері.
5. Підготувати та провести експерименти для порівняння якості роботи розробленої архітектури з існуючими методами класифікації зображень.
6. Розробити інтерфейс програми для демонстрації результатів роботи розробленої архітектури.
7. Проаналізувати результати експериментів та зробити висновки щодо якості роботи розробленої архітектури порівняно з існуючими методами класифікації зображень.

1.5 Специфікація вимог до системи

1.5.1 Функціональні вимоги

1. Програма повинна мати можливість завантаження зображень для наступної класифікації.
2. Програма повинна забезпечувати можливість вибору алгоритму класифікації зображень.
3. Застосунок повинен дозволяти користувачеві налаштовувати параметри алгоритму класифікації зображень.

4. Програма повинна мати можливість відображення результату класифікації на екрані пристрою.
5. Програма повинна дозволяти зберігати результати класифікації у файлову систему пристрою.

1.5.2 Нефункціональні вимоги

1. Програма має бути кросплатформною і працювати на операційних системах Windows, macOS та Linux.
2. Програма повинна забезпечувати високу точність класифікації зображень.
3. Додаток повинен працювати із зображеннями різних розмірів та розширень.
4. Додаток повинен забезпечувати швидкодію під час класифікації зображень.
5. Додаток повинен забезпечувати простий і зрозумілий інтерфейс користувача.

1.5.3 Вимоги до взаємодії з іншими системами

1. Програма повинна мати можливість працювати з файловою системою пристрою для збереження результатів класифікації.
2. Програма повинна забезпечувати взаємодію з операційною системою пристрою для завантаження та відображення зображень.
3. Додаток повинен мати можливість працювати з бібліотеками машинного навчання для реалізації алгоритму класифікації зображень.

1.6 Глосарій

Класифікація зображень: завдання машинного навчання, що полягає у визначенні, якому класу (категорії) належить це зображення.

Нейронна мережа: алгоритм машинного навчання, побудований на основі моделі нейронів, що імітують роботу мозку.

Трансформер (Transformer): архітектура нейронної мережі, яка була представлена у 2017 році у статті "Attention Is All You Need" [24] компанією Google. Вона була розроблена для обробки послідовностей, таких як тексти, але пізніше була успішно застосована для обробки зображень.

Візуальний трансформер (Visual Transformer): модель глибокого навчання, запропонована у 2021 році, заснована на механізмі уваги, що використовується для обробки зображень.

Згорткова нейронна мережа (Convolutional Neural Network, CNN): вид нейронної мережі, який використовується для аналізу зображень та відео.

Сегментація зображень (image segmentation): задача машинного навчання, що полягає у призначенні пікселям зображення певних класів або областей, що дозволяє точно визначити форму та розташування об'єктів на зображенні.

Препроцесинг (preprocessing): процес підготовки даних перед використанням їх для навчання моделі. В контексті класифікації зображень з використанням візуального трансформера це може включати такі кроки, як зміна розмірів зображень, нормалізація значень пікселів, вирівнювання даних тощо.

Повнозв’язне прогнозування (Dense prediction): є сім'єю основних проблем у комп'ютерному зорі, які вивчають відображення від вхідних зображень до складних вихідних структур, включаючи семантичну сегментацію, оцінювання глибини та виявлення об'єктів, серед багатьох інших. У таких завданнях необхідне маркування пікселів на рівні деталізації.

Батч (batch): група прикладів даних, які обробляються одночасно в ході навчання нейронної мережі.

Функція втрат (loss function): функція, яка вимірює різницю між передбаченими і правильними значеннями завдання навчання з учителем.

Збереження моделі (model checkpoint): збережений стан моделі у певний момент часу під час її навчання, який можна використовувати для відновлення навчання або застосування моделі до нових даних.

Гіперпараметри (hyperparameters): налаштування моделі машинного навчання, що визначає її архітектуру, оптимізатор, функцію втрат та інші параметри, які не можуть бути вивчені з даних безпосередньо.

Аугментація даних (data augmentation): техніка, яка використовується для розширення навчального набору даних шляхом застосування різноманітних перетворень до зображень, таких як обертання, зміщення, збільшення / зменшення масштабу тощо. Це дозволяє покращити роботу моделі та зменшити перенавчання.

Точність (accuracy): метрика, яка вимірює відсоток правильних відповідей моделі на задачі класифікації.

Перенесення навчання (transfer learning): метод машинного навчання, за якого попередньо навчена модель використовується як основа для вирішення нового завдання без повного навчання з нуля.

2 МАТЕМАТИЧНІ ОСНОВИ ВІЗУАЛЬНОГО ТРАНСФОРМЕРА

2.1 Трансформери vs CNN

У цьому розділі буде розглянуто огляд сучасних методів класифікації зображень, включаючи нейронні згорткові мережі та методи на основі візуального трансформера.

Згорткові нейронні мережі (CNN) були спочатку розроблені для вирішення задач обробки зображень, і в даний час є найбільш поширеним методом класифікації зображень. CNN складається з декількох шарів, кожен з яких виконує певну операцію, таку як згортка, пониження розмірності, нормалізація і т.п.. Кожен шар може містити декілька фільтрів, які застосовуються до зображення, щоб отримати карти ознак. Після обробки зображення проходить через кілька повнозв'язних шарів, які остаточно виконують класифікацію.

Однак, згорткові нейронні мережі можуть зіткнутися з проблемою обмеженої здатності сприйняття далеких залежностей у зображенні та можуть неефективно працювати із зображеннями, що містять довгі геометричні структури, такі як довгі трубопроводи або шнури.

Замість використання згорткових шарів, як у CNN, візуальний трансформер використовує трансформерні блоки, щоб витягувати ознаки з зображень. Трансформерні блоки обробляють зображення як послідовності, де кожен піксель є вектором ознак. Такий підхід дозволяє моделі візуального трансформера ефективніше працювати з довгими залежностями у зображеннях, а також ефективніше використовувати багатоканальні зображення.

Візуальний трансформер може мати більш високу точність класифікації при роботі з великими зображеннями та/або завданнями, що вимагають аналізу деталей, порівняно із згортковими нейронними мережами. Візуальний трансформер, як і нейронні згорткові мережі, є глибокою нейронною мережею.

Основною перевагою візуального трансформера є його здатність моделювати дальнодіючі взаємодії між елементами зображення на відміну від нейронних

згорткових мереж, які застосовують локальні операції на піксельному рівні. Візуальний трансформер може аналізувати не тільки сусідні області зображення, а й ті, які розташовані далеко, що може покращити якість класифікації.

До недоліків візуального трансформера можна віднести необхідність наявності багатьох тренувальних даних, високої обчислювальної потужності та великої кількості пам'яті.

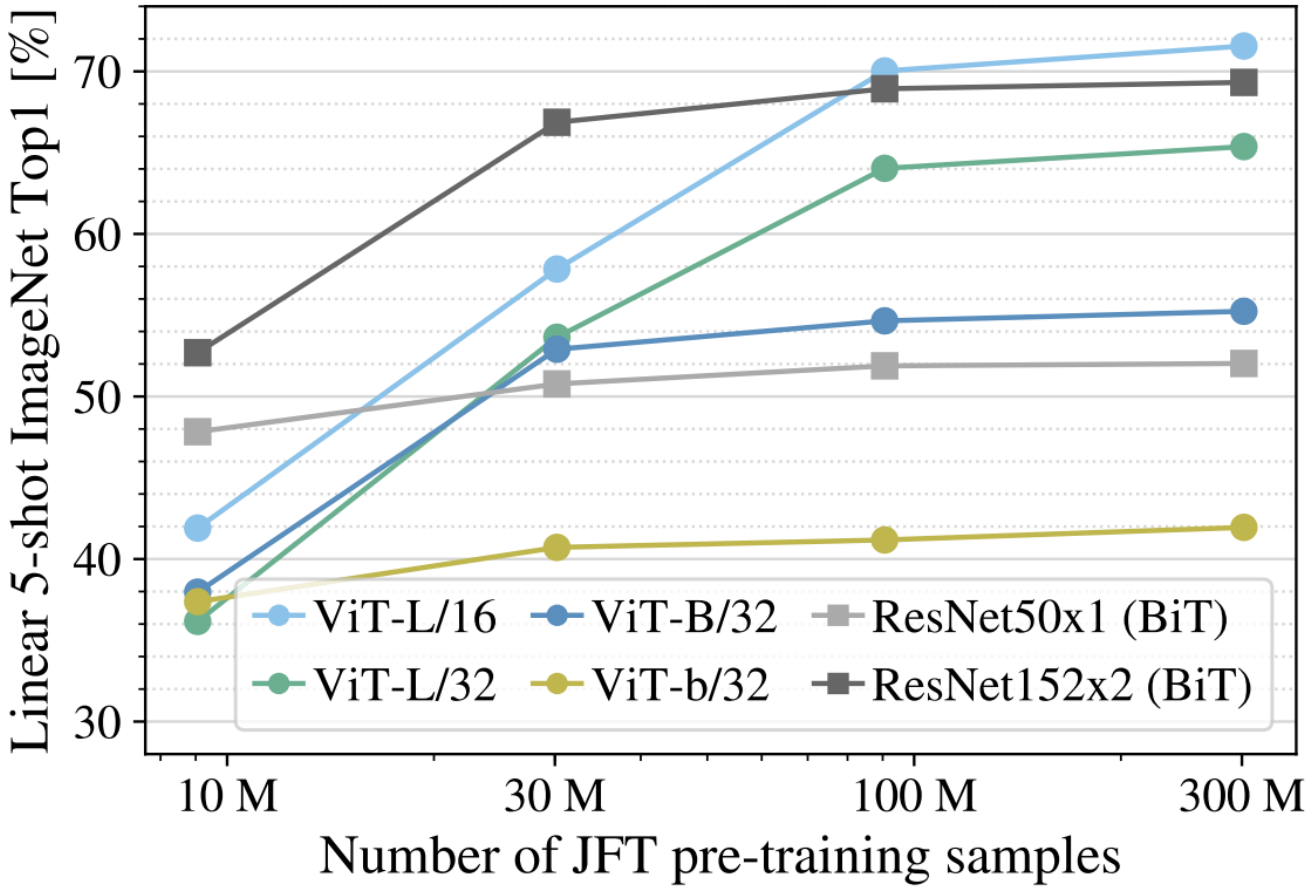


Рисунок 2.1 – Порівняння результатів точності нейронних мереж для класифікації зображень [1]

На рис. 2.1 бачимо графік, який порівнює точність класифікації на наборі даних ImageNet в залежності від кількості зображень тренувальному наборі. Спочатку використовувалося 10 мільйонів зображень. Навіть великі архітектури візуальних трансформерів не можуть зрівнятися з продуктивністю ResNet50, яка має значно меншу кількість параметрів при використанні 10 мільйонів зображень.

Найбільша модель візуального трансформера досягає продуктивності моделі ResNet152 тільки за умови використання 100 мільйонів зображень для попереднього навчання.

З цього можемо зробити висновки, що мережі ResNet показують кращі результати з меншими наборами даних для попереднього навчання, але досягають «плато» раніше, ніж ViT, який показує кращі результати з більшим набором тренувальних даних.

Існують також покращені модифікації трансформерів. Замість звичайної токенізації, що використовується у базовій моделі ViT [1], автори дослідження [25] пропонують прогресивний модуль токенізації для об'єднання сусідніх токенів у один токен (Tokens-to-Token (T2T)), який може моделювати локальну структурну інформацію навколишніх токенів та ітеративно зменшувати довжину токенів. На основі модуля T2T автори розробили модель Tokens-to-Token Vision Transformer (T2T-ViT), яка показує високу продуктивність, коли навчається з нуля на ImageNet, та є більш легкою ніж звичайний ViT (рис. 2.2).

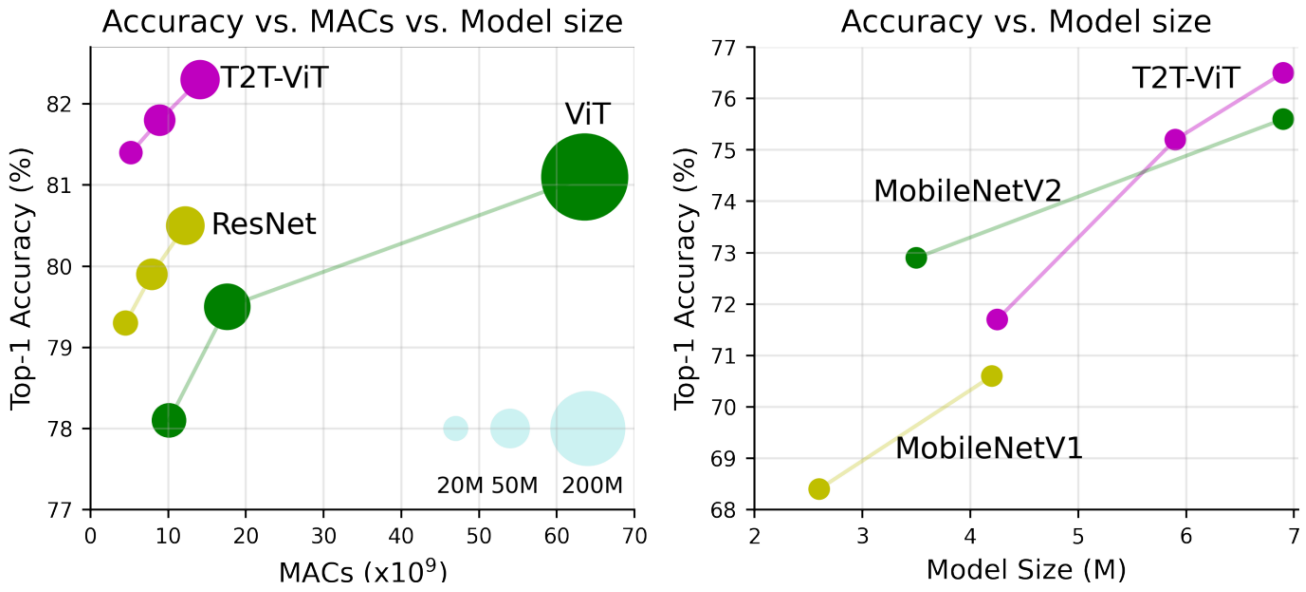


Рисунок 2.2 – Порівняння між T2T-ViT, ViT, ResNets та MobileNets при навчанні з нуля на наборі даних ImageNet [25]

На (рис 2.2) ліворуч наведено криву продуктивності MAC (Multiply–accumulate operation). Праворуч відображено розмір моделі відносно точності. Як можна побачити, T2T-ViT з 21,5 млн параметрів і 4,8 млрд MAC досягає точності top-1 81,5% на ImageNet, що значно вище, ніж у ViT [1] з 48,6 млн параметрів і 10,1 млрд MAC (78,1%). Цей результат також вищий, ніж той, який можна досягнути популярними згортковими нейронними мережами подібного розміру, таких як ResNet50 з 25,5 млн параметрів (76%-79%).

Таким чином, візуальний трансформер являє собою новий підхід до класифікації зображень, який може мати більш високу точність і здатність аналізувати зображення з більш високою деталізацією.

2.2 Опис методу візуального трансформера

Візуальний трансформер складається з двох частин: енкодера (encoder) та декодера (decoder). Енкодер перетворює зображення на послідовність ознак, які потім подаються на вхід трансформеру. Трансформер обробляє послідовність ознак, використовуючи механізми уваги (attention) для виділення найважливіших ознак. На виході формується вектор ознак, що потім подається на вхід декодеру. Декодер перетворює вектор ознак на прогнозування класу зображення (рис. 2.3).

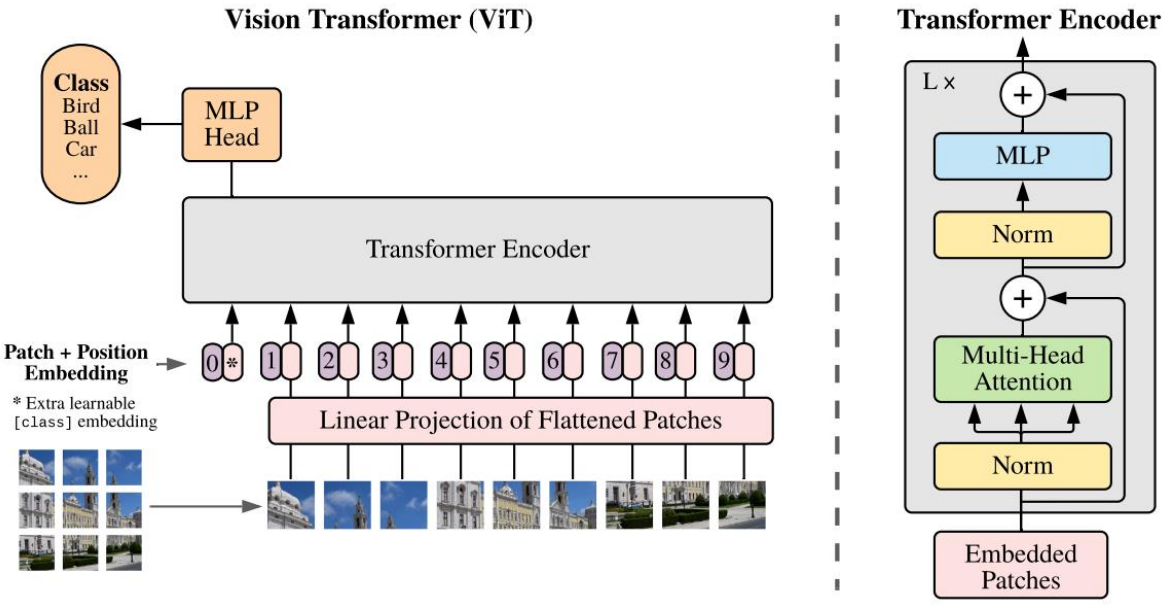


Рисунок 2.3 – Огляд моделі трансформера [1]

Зображення розбивається на фіксовані фрагменти (патчі) однакового розміру, потім відбувається лінійне перетворення, яке трансформує кожний патч у вектор із урахуванням його позиції. Отримана послідовність векторів подається на вхід стандартного енкодера. Класифікація виконується із додаванням «токену класифікації», що навчається, в послідовність [1].

Також цікавим підходом є модель Swin Transformer [4], яка побудована за допомогою ієрархічних карт ознак і має лінійну обчислювальну складність для розміру зображення. У статті [4] автори розширюють застосовність трансформеру таким чином, щоб він міг бути універсальною основою для моделей комп'ютерного зору, як це робиться для обробки природньої мови (NLP), а також, як це роблять згорткові нейронні мережі (CNN) у зоровому сприйнятті.

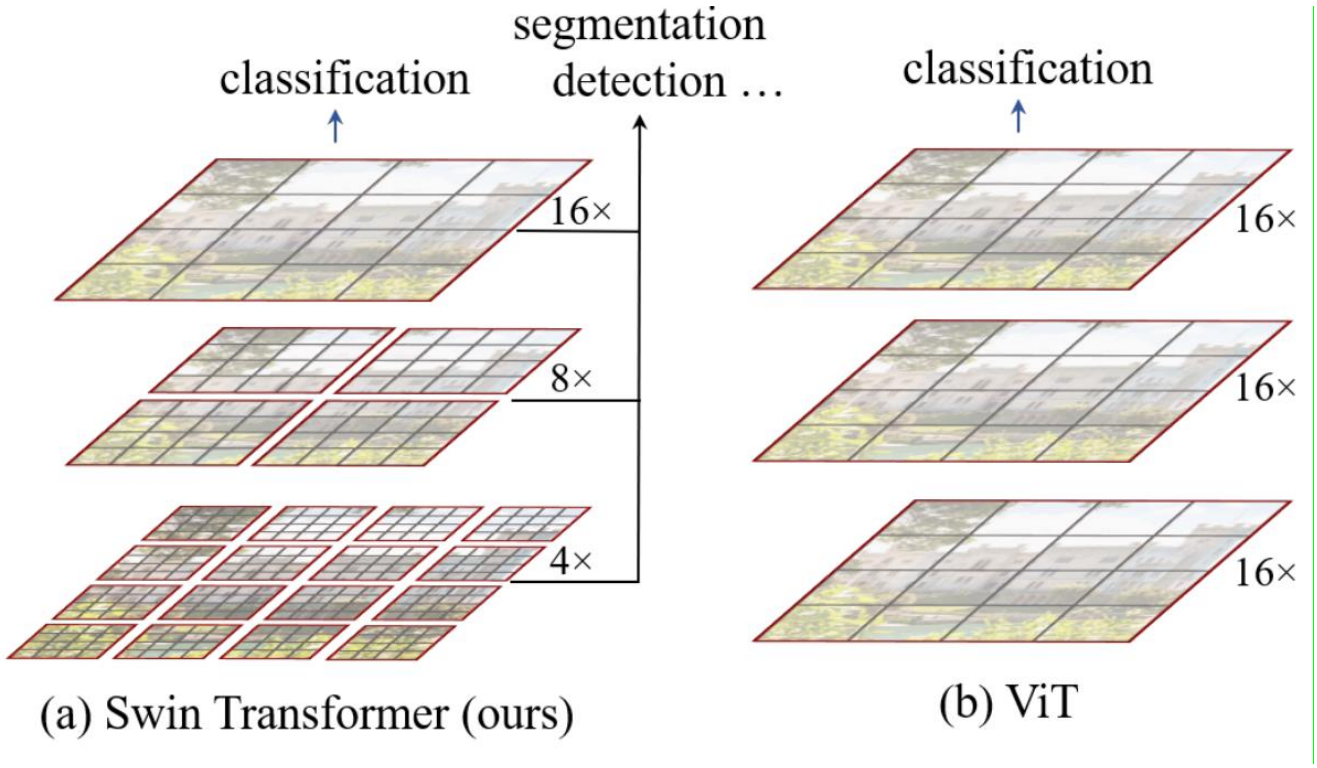


Рисунок 2.4 – Порівняння етапів обробки Swin Transformer та ViT [4]

Swin Transformer будує ієрархічні карти ознак, об'єднавши патчі зображення (показані сірим кольором) (рис. 2.4) на глибших шарах і має лінійну обчислювальну складність щодо розміру вхідного зображення завдяки обчисленням self-attention тільки у межах кожного локального вікна (показаного червоним кольором). Таким

чином, він може служити універсальною основою як для класифікації зображень, так і для завдань розпізнавання. У порівнянні з цим, інші ViT створюють карти ознак лише одного низького розширення і мають квадратичну складність обчислення відносно розміру вхідного зображення через глобальні обчислення self-attention.

Завдяки цим ієрархічним картам ознак, модель Swin Transformer може зручно використовувати методи повнозв'язної класифікації та сегментації, такі як мережі пірамідальних ознак (FPN) [27] або U-Net [28].

Також можна згадати архітектуру CrossViT [26] (рис. 2.5), яка використовує механізм перехресного виклику (cross-attention) та багатомасштабного підходу, щоб поліпшити здатність ViT-моделі обробляти інформацію на різних рівнях деталізації. Ця модель містить стек K мультимасштабних трансформерних енкодерів, кожен з яких використовує дві різні «гілки» для обробки токенів зображень різного розміру і об'єднання токенів в кінці за допомогою ефективного модуля, заснованого на взаємній увазі CLS-токенів.

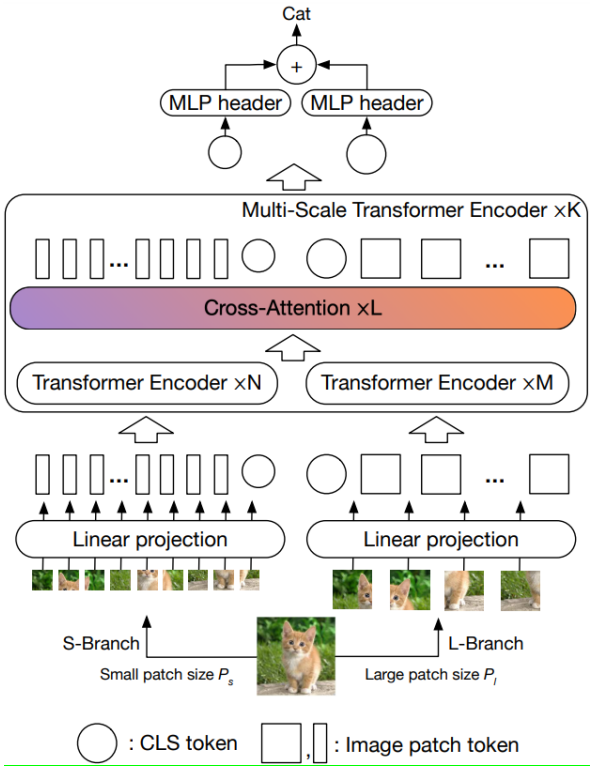


Рисунок 2.5 – Огляд моделі CrossViT [26]

Основною перевагою візуального трансформера є можливість обробки зображень будь-якого розміру без їх попередньої обробки. Крім того, візуальний трансформер дозволяє ефективно використовувати інформацію з великих наборів даних, що робить його особливо корисним для класифікації завдань зображень.

Для навчання візуального трансформера використовується метод зворотного розповсюдження помилки (backpropagation) з використанням функції втрат, наприклад крос-ентропії. Оптимізація здійснюється за допомогою алгоритмів градієнтного спуску, наприклад Adam або SGD.

Для реалізації візуального трансформера використовуються бібліотеки глибокого навчання, такі як PyTorch або TensorFlow, які дозволяють легко створювати та навчати нейронні мережі на основі трансформерів.

2.3 Механізми уваги в методі візуального трансформера

Механізм самоуваги є важливою складовою трансформерної архітектури, яка допомагає моделі ViT знаходити довгострокові залежності та контекстуальну інформацію у вхідних даних. Він дозволяє моделі зосереджуватись на різних областях вхідних даних в залежності від їх важливості для конкретної задачі (рис. 2.6).

Для цього механізм самоуваги обчислює зважену суму вхідних даних, при цьому ваги визначаються подібністю ознак. Це дозволяє моделі надавати більшу вагу важливим ознакам вхідних даних, що сприяє отриманню більш інформативних представлень.

Загалом, механізм самоуваги є обчислювальним інструментом, який допомагає кількісно описувати взаємодії між елементами і допомагає мережі вивчати ієрархічні залежності та вирівнювання вхідних даних. Цей метод довів свою ефективність як ключовий елемент для досягнення більшої стійкості в обробці зображень.

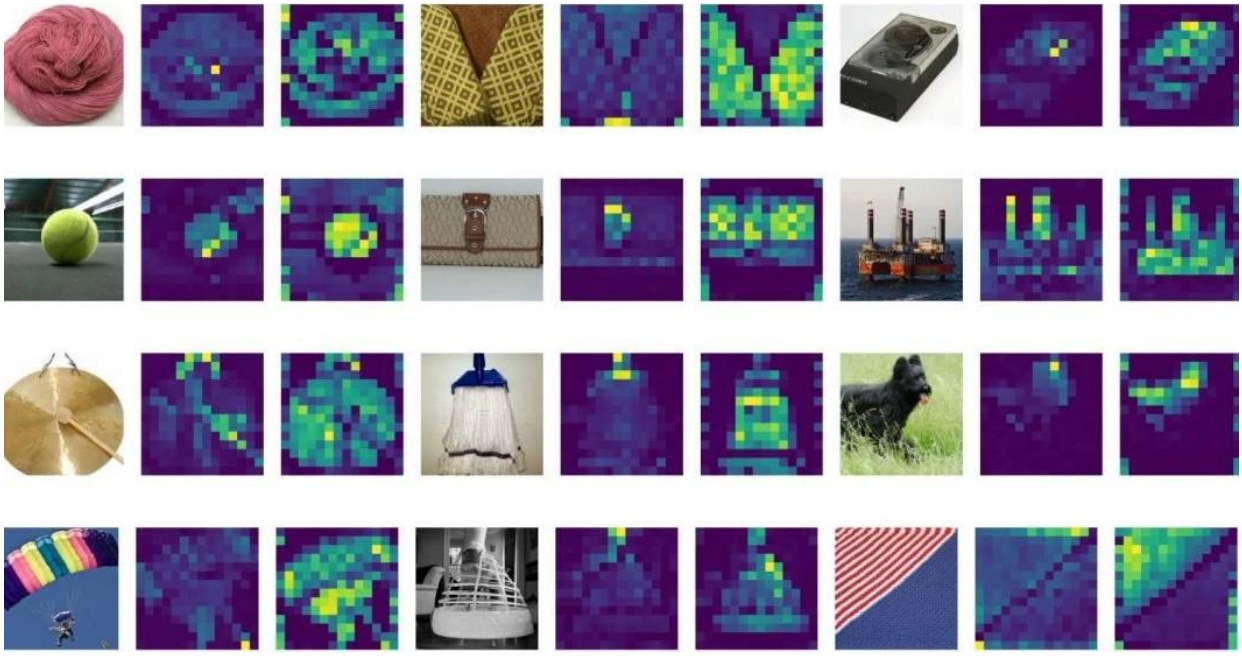


Рисунок 2.6 – Необроблені зображення (ліворуч), карти уваги моделі ViT-S/16 [10] з урахуванням оптимізації різкості (праворуч) та без неї (у середині)

2.4 Карти уваги в візуальному трансформері

Карти уваги (attention maps) візуалізують важливість різних областей вхідного зображення для моделі Vision Transformer. У ViT зображення розбивається на фрагменти, які не перетинаються, потім перетворюються в одновимірний вектор і проходять через кодувальник трансформера.

Карти уваги показують важливість кожного фрагменту зображення відносно інших фрагментів. Ці карти обчислюються за допомогою механізму самоуваги, де кожен токен звертає увагу на всі інші токени для обчислення зваженої суми їхніх представлень.

Карти уваг можна відобразити у вигляді сітки теплових карт (рис. 2.7), де кожна карта показує вагу уваги між конкретним токеном і всіма іншими токенами. Яскравіший колір в тепловій карті вказує на вищу вагу уваги між відповідними токенами. Аналізуючи карти уваг, можна зрозуміти, які частини зображення є найважливішими для задачі класифікації.

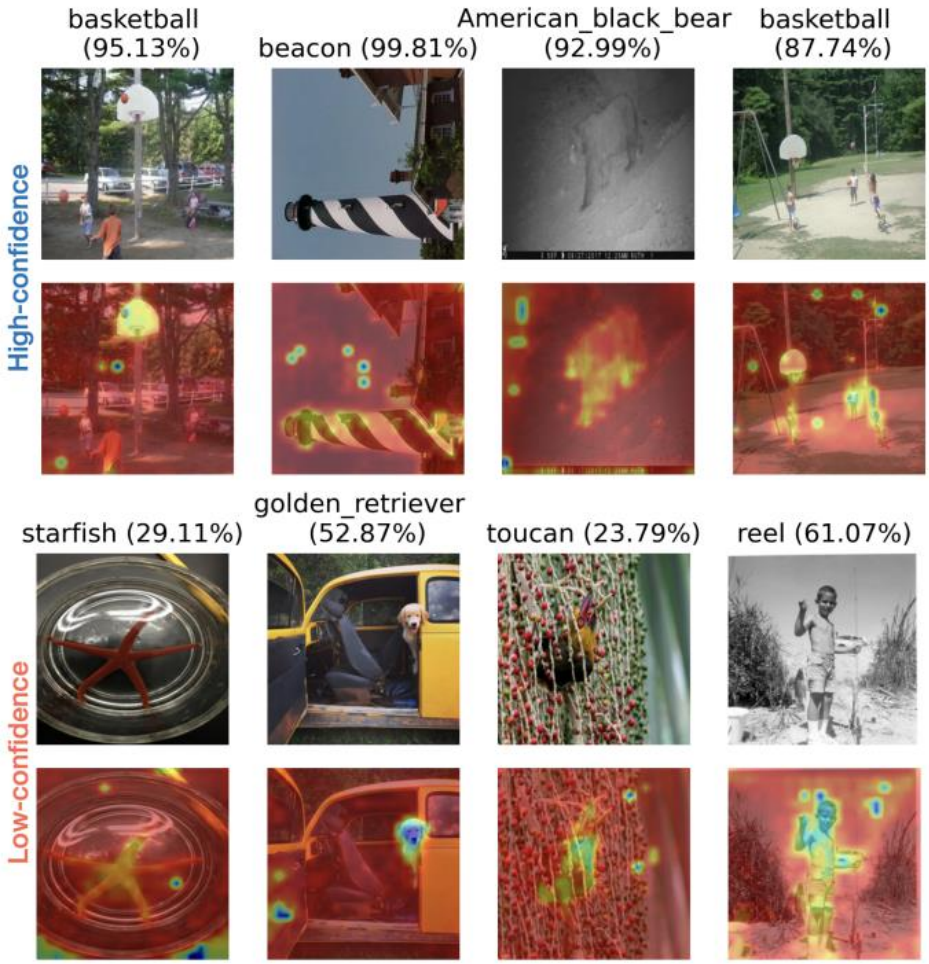


Рисунок 2.7 – Візуалізація карт уваги ViT на зображеннях із ImageNet-A-[11]

3 РОЗРОБКА ЗАСТОСУНКУ ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ВІЗУАЛЬНОГО ТРАНСФОРМЕРА

3.1 Обґрунтування вибору технічних засобів розробки

Застосунок для класифікації зображень за допомогою візуальних трансформерів було розроблено з використанням нижченаведених засобів.

Було використано мову програмування Python. Вона є популярною мовою програмування з великою кількістю бібліотек і фреймворків для роботи з методами штучного інтелекту, машинного навчання, комп'ютерного зору і обробкою зображень. Використання Python дозволяє легко і швидко розробляти комп’ютерні програми для класифікації зображень.

Бібліотека tkinter є стандартною бібліотекою в Python для розробки графічного інтерфейсу користувача. Вона надає можливості для створення вікон, кнопок, полів введення та інших елементів інтерфейсу. Завдяки tkinter додаток може мати зручний інтерфейс для вибору зображень та отримання результатів класифікації.

Бібліотека filedialog є розширенням бібліотеки tkinter, яка містить функції для вибору файлів та папок з діалоговими вікнами. Вона використовується для відкриття діалогового вікна, де користувач може вибрати зображення для обробки.

Бібліотека PIL (Python Imaging Library) надає таку функціональність для роботи з зображеннями як завантаження, збереження, зміна розміру, обрізка та інші операції. Використання PIL дозволяє завантажувати зображення з файлової системи і відображати їх у графічному інтерфейсі додатку.

Бібліотека torch є популярною бібліотекою для роботи з нейронними мережами та глибоким навчанням в Python. Вона містить різні архітектури нейронних мереж, включаючи візуальні трансформери, а також набір функцій для обробки зображень та класифікації. Torch дозволяє використовувати готові моделі класифікації та здійснювати класифікацію зображень.

Бібліотека torchvision є розширенням для бібліотеки torch і надає функції для роботи з комп'ютерним зором та обробки зображень. Вона містить набір утиліт для завантаження даних, попередньої обробки зображень, таких як зміна розміру, обрізка та нормалізація, використання популярних моделей класифікації та інших задач комп'ютерного зору. Torchvision спрощує роботу з обробкою зображень та використання моделей.

Бібліотека timm (PyTorch Image Models) дозволяє розширити функціональність torchvision для моделей класифікації зображень. Вона містить реалізації різних архітектур, включаючи візуальні трансформери. Timm дозволяє легко використовувати готові візуальні трансформери для класифікації зображень.

Бібліотека time надає функції для роботи з часом і вимірювання тривалості виконання певних процедур та фрагментів коду. Використовується для вимірювання часу, потрібного для класифікації зображення.

Бібліотека datetime надає класи та функції для роботи з датами і часом. Використовується для отримання поточного часу та дати для збереження історії класифікації.

У застосунку використовуються популярні моделі класифікації, такі як vit_base_patch16_224, ResNet-152 і InceptionV3. Ці моделі мають попередньо навчені ваги на великих наборах даних, що дозволяє досягти високої точності класифікації зображень. Використання готових моделей дозволяє уникнути навчання мереж «з нуля», що потребує значних обчислювальних потужностей, спрощує процес розробки та забезпечує хорошу якість класифікації.

Файл з класами «imagenet_classes.txt» використовується для зчитування назв класів, що присутні в датасеті ImageNet, для подальшої інтерпретації результатів класифікації. Цей набір даних використовується для тренування багатьох моделей класифікації зображень і містить або одну тисячу різних класів, або близько 21 тисячі класів для розширеної версії датасету ImageNet.

Файл з історією "history.txt" використовується для запису результатів класифікації та подальшого зчитування.

Загалом, вибір цих технічних засобів дозволяє зручно розробити застосунок для класифікації зображень з використанням візуального трансформера. Мова програмування Python та бібліотеки tkinter, PIL, torch, torchvision та timm надають потужні інструменти для роботи з зображеннями, моделями класифікації та розробки графічного інтерфейсу користувача. Готові моделі класифікації, такі як vit_base_patch16_224 [3], ResNet-152 [15] та InceptionV3 [16] забезпечують високу точність класифікації завдяки попередньому навчанню на великих наборах даних, зокрема на ImageNet.

Такий набір технічних засобів дозволяє зручно вибирати зображення, візуалізувати їх у графічному інтерфейсі, класифікувати за допомогою візуального трансформера та відображати результати класифікації. Це робить додаток ефективним і зручним інструментом для розпізнавання об'єктів на зображеннях.

Вибір таких технічних засобів обґрунтований їх потужністю, широким спектром можливостей та підтримкою популярних моделей класифікації зображень.

3.2 Архітектура програми

Архітектура застосунку базується на об'єктно-орієнтованому підході і має структуру, описану нижче.

Класс ImageClassifierGUI відповідає за створення та управління графічним інтерфейсом користувача. Він містить методи для відображення зображення, вибору моделі, класифікації зображення та відображення результатів.

Функція init() (рис. 3.1) ініціалізує об'єкт класу ImageClassifierGUI та створює головне вікно програми. В ньому відбувається налаштування елементів інтерфейсу, встановлення значень за замовчуванням та виклик методів для відображення та класифікації зображення.

```python
def __init__(self, master):
    self.master = master
    master.title("Image Classifier")

    self.image_path = "saint_bernard.jpg"

    self.instruction_button = tk.Button(master, text="Instruction", command=self.open_instruction_window)
    self.instruction_button.pack(side=tk.TOP, fill=tk.X)

    self.examples_label = tk.Label(master, text="Examples")
    self.examples_label.pack()
    self.examples = ["Saint Bernard", "Squirrel", "Rabbit"]
    self.examples_selection = tk.StringVar()
    self.examples_selection.set(self.examples[0])
    self.examples_dropdown = tk.OptionMenu(master, self.examples_selection, *self.examples, command=self.choose_example)
    self.examples_dropdown.pack(fill=tk.X)

    self.image_label = tk.Label(master)
    self.image_label.pack(pady=10)

    self.time_label = tk.Label(master, text="")
    self.time_label.pack()

    self.choice_model_label = tk.Label(master, text="Choice Model:")
    self.choice_model_label.pack(anchor=tk.W)

    self.model_selection = tk.StringVar()
    self.model_selection.set("vit_base_patch16_224")

    self.model_options = ["vit_base_patch16_224", "ResNet-152", "InceptionV3"]
    for model in self.model_options:
        tk.Radiobutton(master, text=model, variable=self.model_selection, value=model).pack(anchor=tk.W)

    self.choose_image_button = tk.Button(master, text="Choose Image", command=self.choose_image)
    self.choose_image_button.pack(side=tk.TOP, fill=tk.X)

    self.classify_button = tk.Button(master, text="Classify Image", command=self.classify_image)
    self.classify_button.pack(pady=10, side=tk.TOP, fill=tk.X)

    self.results_label = tk.Label(master, text="Results:", anchor=tk.CENTER)
    self.results_label.pack(fill=tk.X)

    self.results_table = ttk.Treeview(master, columns=("Number", "Index", "Class", "Accuracy"), show="headings")
    self.results_table.heading("Number", text="#")
    self.results_table.heading("Index", text="Index")
    self.results_table.heading("Class", text="Class")
    self.results_table.heading("Accuracy", text="Accuracy")
    self.results_table.column("Number", width=25)
    self.results_table.column("Index", width=40)
    self.results_table.column("Class", width=215)
    self.results_table.column("Accuracy", width=70)
    self.results_table.pack(fill=tk.X)

    self.classification_results = tk.Label(master, text="")
    self.classification_results.pack()

    self.load_history_button = tk.Button(master, text="History", command=self.show_history)
    self.load_history_button.pack(side=tk.BOTTOM, fill=tk.X)

    self.classify_image()
    self.show_image()
```

Рисунок 3.1 – Функція для створення головного вікна програми

Функція choose_image() (рис. 3.2) викликає діалогове вікно для вибору зображення з файлової системи. Обране зображення після вибору відображається в графічному інтерфейсі.

```python
def choose_image(self):
    self.image_path = filedialog.askopenfilename(title="Choose an image", filetypes=[("Image Files", "*.jpg *.png *.jpeg")])
    self.show_image()
    self.classify_image()
```

Рисунок 3.2 – Функція для вибору зображення

Функція show_image() (рис. 3.3) завантажує зображення з вказаного шляху та змінює його розмір, щоб відповідати вимогам моделі класифікації. Потім зображення відображається в графічному інтерфейсі.

```
def show_image(self):
    if self.image_path:
        image = Image.open(self.image_path)
        image = image.resize((300, 300))
        photo = ImageTk.PhotoImage(image)
        self.image_label.configure(image=photo)
        self.image_label.image = photo
```

Рисунок 3.3 – Функція для завантаження та зміни розміру зображення

Функція choose_example() (рис. 3.4) обробляє вибір користувача зі списку вбудованих прикладів зображень. Функція змінює шлях зображення відповідно до обраного прикладу та оновлює відображення та класифікацію.

```
def choose_example(self, example):
    if example == "Saint Bernard":
        self.image_path = "saint_bernard.jpg"
    elif example == "Squirrel":
        self.image_path = "squirrel.jpg"
    elif example == "Rabbit":
        self.image_path = "rabbit.jpg"
    self.show_image()
    self.classify_image()
```

Рисунок 3.4 – Функція для вибору зображення із прикладів

Функція classify_image() викликає обрану модель класифікації (рис. 3.5) (залежно від вибору користувача) та виконує класифікацію зображення. Виконується обробка зображення, застосовуючи необхідні перетворення (зміна

розміру, центральне обрізання, перетворення в тензор, нормалізація значень пікселів) (рис. 3.6), після чого виконується класифікація зображення за допомогою обраної моделі. Після отримання результатів класифікації, метод відображає їх у графічному інтерфейсі.

```
model_name = self.model_selection.get()

if model_name == "vit_base_patch16_224":
    model = timm.create_model(model_name, pretrained=True)
elif model_name == "ResNet-152":
    model = models.resnet152(pretrained=True)
elif model_name == "InceptionV3":
    model = models.inception_v3(pretrained=True)
else:
    raise ValueError("Unknown model")
```

Рисунок 3.5 – Виклик обраної моделі класифікації

```
image = Image.open(self.image_path).convert('RGB')

transform = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])

image = transform(image).unsqueeze(0)
```

Рисунок 3.6 – Обробка зображення з застосуванням необхідних перетворень

Функція save_to_history() (рис. 3.7) зберігає результати класифікації у файлі "history.txt". Вона отримує дані про час, назву зображення, використану модель, витрачений час, клас та точність класифікації.

```python
def save_to_history(self, result):
    timestamp = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
    image_name = self.image_path.split("/")[-1]
    model_name = self.model_selection.get()
    time_taken = self.time_label.cget("text").split(":")[1].strip()
    class_name = result[2]
    accuracy = f"{result[3]:.3f}"

    with open("history.txt", "a") as f:
        f.write(f"{timestamp},{image_name},{model_name},{time_taken},{class_name},{accuracy}\n")
```

Рисунок 3.7 – Функція для збереження результатів класифікації

Функція show_history() (рис. 3.8) відкриває окреме вікно зі списком історії класифікації. Вона читає дані з файлу "history.txt" і відображає їх у вигляді таблиці, використовуючи бібліотеку ttk.Treeview.

```python
def show_history(self):
    history_window = tk.Toplevel(self.master)
    history_window.title("History")
    history_table = ttk.Treeview(history_window, columns=("Date&Time", "Image", "Model", "Time", "Class", "Accuracy"), show="headings")

    history_table.heading("Date&Time", text="Date&Time")
    history_table.heading("Image", text="Image")
    history_table.heading("Model", text="Model")
    history_table.heading("Time", text="Time")
    history_table.heading("Class", text="Class")
    history_table.heading("Accuracy", text="Accuracy")

    history_table.column("Date&Time", width=120)
    history_table.column("Image", width=200)
    history_table.column("Model", width=125)
    history_table.column("Time", width=80)
    history_table.column("Class", width=200)
    history_table.column("Accuracy", width=70)

    history_table.pack(fill=tk.BOTH, expand=True)

    with open("history.txt", "r") as f:
        for line in f:
            parts = line.strip().split(",")
            if len(parts) >= 6:
                history_table.insert("", tk.END, values=(parts[0], parts[1], parts[2], parts[3], parts[4], parts[5]))
            else:
                print("Invalid data format in history.txt")
```

Рисунок 3.8 – Функція для перегляду історії класифікації

Функція run() (рис. 3.9) запускає головний цикл графічного інтерфейсу, який дозволяє взаємодіяти з програмою та виконувати дії користувача.

```python
def run(self):
    self.master.mainloop()
```

Рисунок 3.9 – Функція запуску головного циклу графічного інтерфейсу

Така архітектура програми дозволяє зручно управляти відображенням зображень, вибором моделі та класифікацією, забезпечуючи користувачеві зручний та інтуїтивно зрозумілий інтерфейс та перегляд історії класифікації.

3.2 Розробка інтерфейсу додатка

Для роботи програмного застосунку були використані зображення (рис. 3.10), зроблені самостійно за допомогою телефону та знімку екрану комп'ютера при перегляді різних відео.

Рисунок 3.10 – Приклади зображень із власного набору

Інтерфейс розробленого додатку містить інструкцію для користувача, приклади, які можна обрати з випадаючого списку (зображення, які містять сенбернара, білку та кролика), зображення, час за який було класифіковано зображення, можливість вибору моделі, можливість вибору власного зображення, можливість розпочати класифікацію зображення, таблицю результатів вже класифікованого зображення та можливість переглянути історію класифікацій (рис. 3.11).

Рисунок 3.11 – Інтерфейс додатку на момент запуску

На (рис. 3.12) та (рис. 3.13) наведено результати класифікації зображення за допомогою згорткових нейронних мереж ResNet та InceptionV3 відповідно.

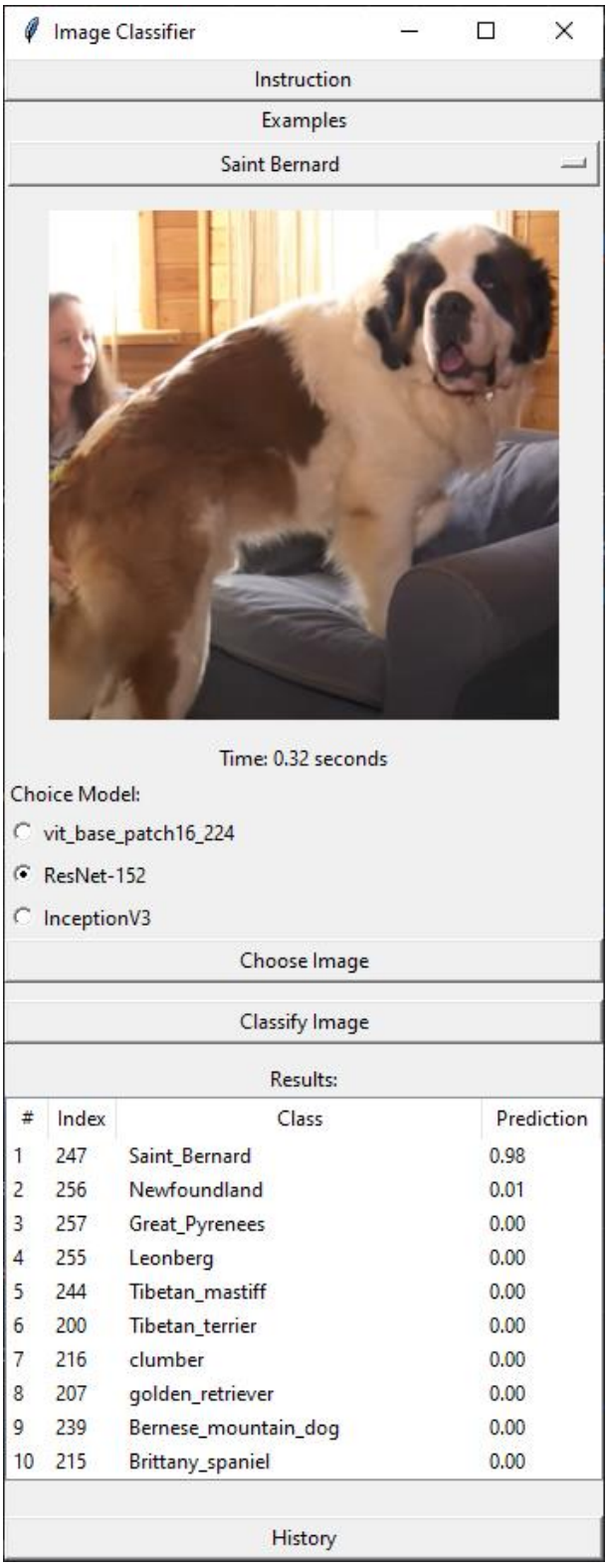


Рисунок 3.12 – Класифікація зображення за допомогою ResNet-152

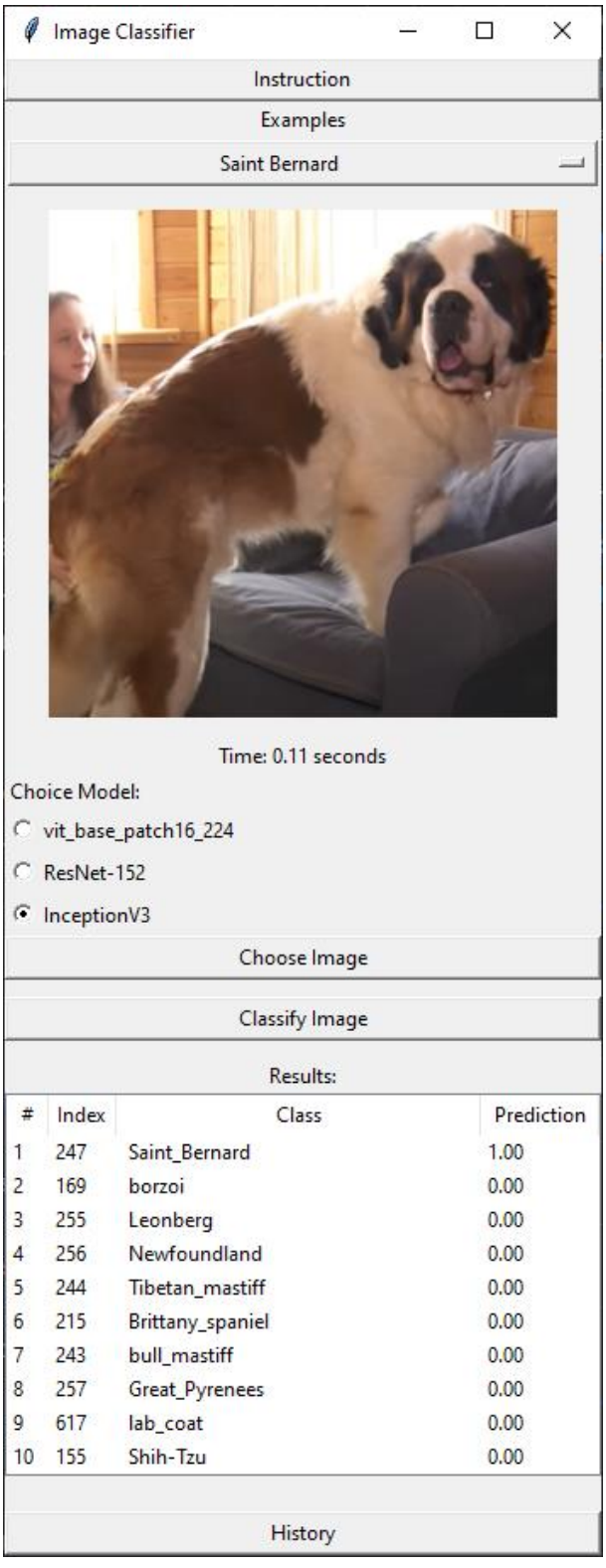
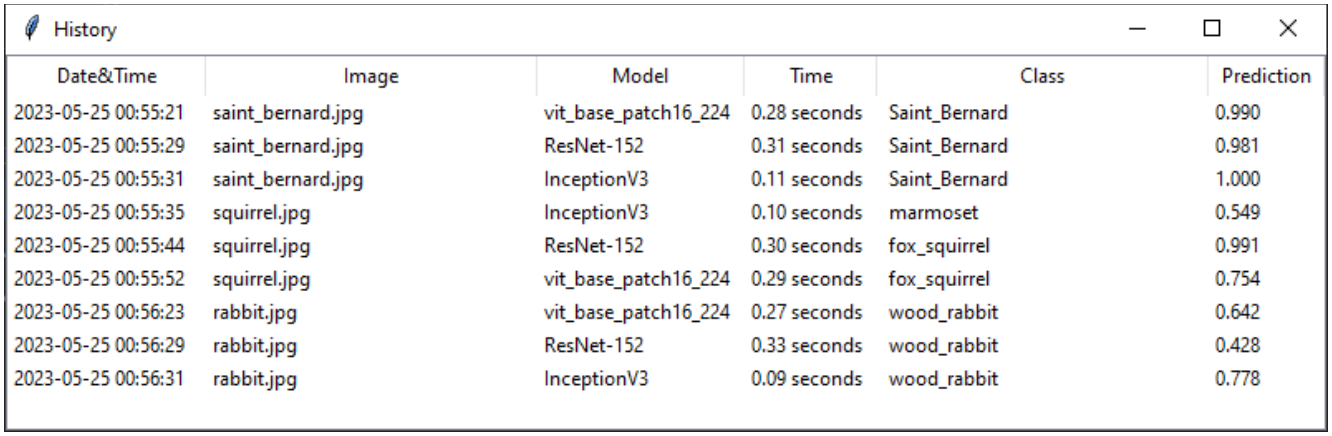


Рисунок 3.13 – Класифікація зображення за допомогою InceptionV3

При натисканні кнопки «History» відкривається вікно з таблицею. Перший стовпчик таблиці «Date&Time» відображає дату та час, коли було класифіковано зображення. Другий стовпчик таблиці «Image» відображає назву зображення. Третій стовпчик таблиці «Model» відображає модель, за допомогою якої проводилась класифікація зображення. Четвертий стовпчик таблиці «Time» відображає час, за який було класифіковано зображення. П’ятий стовпчик таблиці «Class» відображає до якого класу належить зображення. Шостий стовпчик таблиці «Prediction» показує результат класифікації (рис. 3.14).



Date&Time	Image	Model	Time	Class	Prediction
2023-05-25 00:55:21	saint_bernard.jpg	vit_base_patch16_224	0.28 seconds	Saint_Bernard	0.990
2023-05-25 00:55:29	saint_bernard.jpg	ResNet-152	0.31 seconds	Saint_Bernard	0.981
2023-05-25 00:55:31	saint_bernard.jpg	InceptionV3	0.11 seconds	Saint_Bernard	1.000
2023-05-25 00:55:35	squirrel.jpg	InceptionV3	0.10 seconds	marmoset	0.549
2023-05-25 00:55:44	squirrel.jpg	ResNet-152	0.30 seconds	fox_squirrel	0.991
2023-05-25 00:55:52	squirrel.jpg	vit_base_patch16_224	0.29 seconds	fox_squirrel	0.754
2023-05-25 00:56:23	rabbit.jpg	vit_base_patch16_224	0.27 seconds	wood_rabbit	0.642
2023-05-25 00:56:29	rabbit.jpg	ResNet-152	0.33 seconds	wood_rabbit	0.428
2023-05-25 00:56:31	rabbit.jpg	InceptionV3	0.09 seconds	wood_rabbit	0.778

Рисунок 3.14 – Інтерфейс вікна історії

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Показники точності

Для порівняння результатів класифікації зображень з використанням візуального трансформера та інших методів (ResNet-152 та InceptionV3) було використано наступні показники точності.

Точність (accuracy) (4.1) є основним показником, що використовуються для оцінки якості класифікації. Вона вимірює відношення кількості правильно класифікованих зображень до загальної кількості зображень у тестовому наборі. Для збалансованих наборів даних, коли кількість зображень кожного класу є приблизно однаковою (як в нашому випадку), точність може бути використана як надійна метрика оцінки ефективності класифікаційної моделі, але в задачах з нерівними кількостями зображень для різних класів ця метрика може викривляти реальні результати.

Математично записати точність можна в наступному вигляді:

$$Accuracy = \frac{\text{Кількість правильно класифікованих зображень}}{\text{Загальна кількість зображень}} \tag{4.1}$$

Влучність класифікації (precision) (4.2) визначає, яка частка зображень, класифікованих як позитивні, насправді є правильно класифікованими. Вона обчислюється як відношення кількості правильно класифікованих позитивних зображень до загальної кількості позитивних зображень:

$$Precision = \frac{\text{Кількість правильно класифікованих позитивних об'єктів}}{\text{Загальна кількість позитивних об'єктів}} \tag{4.2}$$

Позитивними зображеннями в контексті експерименту класифікації вважаються зображення конкретного класу. Нехай ми класифікуємо клас «balloon». Тоді в чисельнику (4.2) буде знаходитися кількість зображень, які були

класифіковані як «balloon» та насправді відносяться до цього класу (true positives). В знаменнику (4.2) до цього значення додається кількість випадків, коли зображення інших класів (не «balloon») були класифіковані як «balloon».

Повнота (recall) (4.3) вимірює, яка частка фактично позитивних зображень була правильно класифікована моделлю. Вона обчислюється як відношення кількості правильно класифікованих позитивних зображень до загальної кількості фактично позитивних зображень:

$$Recall = \frac{\text{Кількість правильно класифікованих позитивних об'єктів}}{\text{Загальна кількість фактично позитивних об'єктів}} \quad (4.3)$$

Якщо ми продовжимо опис прикладу для класу «balloon», то в чисельнику (4.3) буде знаходитися те ж саме, що й для (4.2). Але в знаменнику (4.3) до цього значення додається кількість випадків, коли зображення класу «balloon» були помилково класифіковані як такі, що відносяться до інших класів.

F1-показник (F1-score) (4.4) є одним з часто використовуваних показників точності в задачах класифікації, особливо коли маємо справу з незбалансованими класами. Він представляє собою гармонічне середнє між precision і recall та використовується для оцінки балансу між цими двома показниками. F1-показник набуває значення від 0 до 1, де 1 відповідає ідеальному балансу між precision і recall, а 0 вказує на найгіршу ефективність моделі:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (4.4)$$

F1 є конкретним випадком загального показника F_β – score, який можна записати наступним чином:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Використання показника F_β зі значеннями $\beta = 2,\ \beta = 3$ дозволяє в практичних задач класифікації врахувати те, що ціна помилки класифікації може бути різною. Якщо припустити для прикладу, що класифікатор визначає наявність певної хвороби у пацієнта – ситуація помилкового знаходження хвороби за її відсутності може бути набагато менш небезпечною (чи навпаки – більш небезпечною) порівняно із ситуацією класифікації відсутності хвороби за її наявності. Наприклад, показник F2 вважає значення recall більш важливим за precision. Для наших експериментів будемо вважати важливості помилок рівноцінними і використаємо F1.

Використання цих показників точності дозволяє порівняти ефективність використання візуального трансформера з іншими методами, такими як ResNet-152 та InceptionV3, для вирішення задачі класифікації зображень.

Для проведення експериментальних досліджень та аналізу результатів, були обчислені значення цих показників для кожного методу класифікації, зокрема для візуального трансформера, ResNet-152 та InceptionV3.

4.2 Порівняння результатів класифікації зображень із використанням візуального трансформера та інших методів класифікації

Існує кілька популярних моделей візуальних трансформерів [1] (табл. 4.1). Кожна модель має відмінність в кількості шарів, розмірі прихованого простору D, розмірі багатошарового персептрона MLP, кількості голів самоуваги та загальній кількості параметрів. Ці характеристики визначають специфікації моделі та її потужність для розв'язання різних завдань обробки зображень.

Кількість шарів (Layers) в моделі Vision Transformer вказує на кількість повторюваних блоків, що складають модель.

Прихований розмір D (Hidden size D) в моделі Vision Transformer (ViT) вказує на розмір простору прихованих представлень, що використовується в

трансформері. Цей розмір визначається кількістю прихованих вузлів у кожному шарі моделі. Він впливає на потужність та можливості моделі.

Розмір MLP (MLP size) в моделі Vision Transformer (ViT) вказує на кількість прихованих вузлів у шарі многошарового перцептрона (MLP), який використовується в трансформері.

Кількість голів самоуваги (Heads) в моделі Vision Transformer (ViT) вказує на кількість паралельних голів самоуваги, які використовуються в трансформері для обробки інформації.

Кількість параметрів (Params) в моделі Vision Transformer (ViT) вказує на загальну кількість вагових коефіцієнтів, які підлягають навчанню у моделі.

Таблиця 4.1 – Детальна інформація про конфігурації трьох основних варіантів моделі Vision Transformer [1]

Модель	Кількість шарів	Прихований розмір D	Розмір MLP	Кількість голів самоуваги	Кількість параметрів
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Для класифікації зображень та порівняння результатів було створено власний набір даних (наведений в додатку А). Цей набір складається з 100 зображень розміром 300×300 пікселів і включає 10 збалансованих класів: «aircraft_carrier», «balloon», «castle», «crayfish», «mantis», «pizza», «warplane», «seashore», «shoe_shop», «Pembroke». Варто зазначити, що всі ці класи також присутні в широковживаному наборі даних ImageNet, на основі якого були навчені моделі: «vit_base_patch16_224», «ResNet-152» та «InceptionV3». Також використовувалися два додаткові класи «sphinx_cat», «mouse_animal» по 10 зображень в кожному з них. Варто зазначити, що ці класи не присутні в тренувальному наборі ImageNet.

За результатами, наведеними в табл. 4.2, можна зробити наступні висновки.

Візуальний трансформер (ViT) має точність (0,79) що є трохи нижчою, ніж точність ResNet-152 (0,83), але вищою, ніж точність InceptionV3 (0,61). F1-показник (0,8827) також є високим, що показує добре збалансовану комбінацію точності та повноти.

Таблиця 4.2 – Результати порівняння точності класифікації

	vit_base_patch16_224	ResNet-152	InceptionV3
Accuracy	0,79	0,83	0,61
Precision	0,79	0,83	0,61
Recall	1	1	1
F1-score	0,8827	0,9071	0,7578
Середній час обробки, сек.	0,29	0,32	0,10

ResNet-152 має найвищу точність (0,83) серед усіх моделей, разом з високою точністю прогнозування та повнотою. F1-показник (0,9071) також є високим, що свідчить про його хорошу здатність до класифікації.

InceptionV3 має найнижчу точність (0,61) серед усіх моделей, але він компенсує це швидкістю обробки (середній час 0,10). F1-показник (0,7578) також показує середній рівень точності та повноти.

Оцінювати моделі лише за точністю може бути недостатньо, оскільки інші показники, такі як точність прогнозування, повнота та F1-показник, можуть дати більш повну картину про ефективність класифікації. Також, враховуючи швидкість обробки, можна зробити компроміс між точністю та швидкістю залежно від конкретних вимог проєкту.

Якщо взяти зображення класів, яких немає у навчальному наборі даних, наприклад, зображення котів породи «сфінкс» («sphinx_cat») та зображення миші (тварина) («mouse_animal»), можна побачити з табл. 4.3 – 4.8, що візуальний трансформер, можливо, має трохи вищу узагальнюючу здатність порівняно із іншими моделями.

З 10 зображень класу «sphinx_cat» візуальний трансформер знайшов кота на 4 з них (двічі «Egyptian_cat» та двічі «Siamese_cat»), тоді як ResNet-152 знайшов кота лише на 1 зображенні («Siamese_cat»), а InceptionV3 не знайшов жодного разу (табл. 4.3 – табл. 4.5). Цікавими є результати класифікації зображень 5 та 6, обидва розпізнані всіма трьома моделями як «Italian_greyhound», але ResNet-152 та InceptionV3 приймають набагато більш впевнено рішення щодо цього класу (показник вірогідності від 0.6 до 0.99), в той час як ViT класифікує його з ймовірностями 0.21 та 0.422. Також, половина результатів класифікації для моделей ResNet-152 та InceptionV3 співпадають, але якщо порівняти ViT та ResNet-152, то можна побачити 3 співпадіння, і 2 співпадіння між результатами ViT та InceptionV3. Ці результати підкреслюють якісну відмінність класифікації зображень із використанням трансформера.

Таблиця 4.3 – Класифікація зображень кота «сфінкс» за допомогою ViT

№ зображення	Клас	Вірогідність
1	Egyptian_cat	0.385
2	plastic_bag	0.264
3	Mexican_hairless	0.946
4	Siamese_cat	0.636
5	Italian_greyhound	0.210
6	Italian_greyhound	0.422
7	Italian_greyhound	0.662
8	Siamese_cat	0.952
9	Egyptian_cat	0.267
10	Chihuahua	0.317

Таблиця 4.4 – Класифікація зображень кота «сфінкс» за допомогою ResNet-152

№ зображення	Клас	Вірогідність
1	Chihuahua	0.536
2	Chihuahua	0.418
3	Mexican_hairless	0.967
4	Chihuahua	0.386
5	Italian_greyhound	0.948
6	Italian_greyhound	0.618
7	Chihuahua	0.681
8	Chihuahua	0.720
9	Siamese_cat	0.502
10	Mexican_hairless	0.544

Таблиця 4.5 – Класифікація зображень кота «сфінкс» за допомогою InceptionV3

№ зображення	Клас	Вірогідність
1	Chihuahua	0.671
2	Boston_bull	0.988
3	piggy_bank	0.680
4	Chihuahua	0.833
5	Italian_greyhound	0.771
6	Italian_greyhound	0.997
7	Mexican_hairless	0.915
8	Chihuahua	0.619
9	Chihuahua	1.000
10	basenji	0.988

З 10 зображень класу «mouse_animal» візуальний трансформер класифікував 8 як «mousetrap» (пастка для миші), тоді як ResNet-152 – тільки три зображення як «mousetrap» з десяти, а InceptionV3 «mousetrap» – два з десяти (табл. 4.6 – 4.8).

Таблиця 4.6 – Класифікація зображень миші за допомогою ViT

№ зображення	Клас	Вірогідність
1	mousetrap	0.227
2	weasel	0.270
3	mousetrap	0.506
4	mousetrap	0.482
5	mousetrap	0.281
6	mink	0.145
7	mousetrap	0.681
8	mousetrap	0.288
9	mousetrap	0.937
10	mousetrap	0.265

Таблиця 4.7 – Класифікація зображень миші за допомогою ResNet-152

№ зображення	Клас	Вірогідність
1	hamster	0.453
2	hamster	0.746
3	mousetrap	0.721
4	hamster	0.682
5	mousetrap	0.628
6	fox_squirrel	0.407
7	hamster	0.909
8	wood_rabbit	0.566
9	mousetrap	0.894
10	weasel	0.335

Таблиця 4.8 – Класифікація зображень миші за допомогою InceptionV3

№ зображення	Клас	Вірогідність
1	hamster	0.993
2	hamster	0.999
3	mousetrap	0.995
4	hamster	0.718
5	hamster	0.882
6	fox_squirrel	0.851
7	mousetrap	0.619
8	hamster	0.883
9	fox_squirrel	0.754
10	mink	0.741

Загалом, на основі наданих результатів можна стверджувати, що ResNet-152 є найбільш ефективною моделлю (з обраних та розглянутих в цьому проєкті для цього конкретного датасета) для класифікації зображень, оскільки вона має найвищу точність та F1-показник. Однак, якщо швидкість обробки є критичним фактором, то InceptionV3 може бути більш практичним вибором. Візуальний трансформер (ViT) також має свої переваги, особливо з точки зору узагальнюючої здатності, точності прогнозування та повноти.

Варто також врахувати, що ViT є відносно новою моделлю в активній фазі досліджень, тому в майбутньому він може стати (і, можливо, вже поступово стає) більш конкурентоспроможною альтернативою для використання замість традиційних конволюційних нейронних мереж. Запровадження візуальних трансформерів також може виявитись корисним для інших завдань обробки зображень, таких як детектування об'єктів чи сегментація, оскільки вони можуть бути більш здатні до виявлення контекстуальних залежностей та обробки глобальної інформації.

Також варто зазначити, що ResNet-152 може бути більш ефективним лише для даної задачі класифікації ста зображень. Якщо у проєкті потрібно вирішити інші завдання обробки зображень, такі як детектування об'єктів чи сегментацію, можуть бути використані інші моделі, які будуть більш підходящими для цих конкретних вимог. Наприклад, для детектування об'єктів можуть бути використані моделі, які базуються на архітектурі DETR (Detection Transformer) [22] та ViT-Detection [23], які здатні виявляти та класифікувати об'єкти на зображеннях з високою точністю. У випадку сегментації, моделі, які використовують механізми уваги та декодери засновані на принципах трансформерів, можуть досягти кращих результатів.

Варто зауважити, що вибір моделі не є завжди остаточним навіть для конкретної прикладної задачі. Залежно від умов і особливостей проєкту можна провести додаткові експерименти та налаштування задля досягнення кращих результатів. Розвиток та дослідження використання нейромережевих моделей в галузі комп'ютерного зору активно тривають, і нові архітектури та методи постійно з'являються. Таким чином, завжди існує можливість вдосконалювати результати і знаходити нові найкращі рішення.

ВИСНОВКИ

У рамках виконання дипломного проєкту було розроблено та реалізовано модуль інформаційної системи у вигляді застосунку для проведення класифікації зображень за допомогою візуального трансформера та згорткових нейронних мереж.

Було проведено аналіз предметної області та існуючих розв'язків задачі класифікації зображень.

Розроблено застосунок для класифікації зображень з використанням візуального трансформера та згорткових нейронних мереж.

Проведено експериментальні дослідження та порівняння результатів класифікації зображень з використанням візуального трансформера та інших методів класифікації.

Отже, на основі проведених досліджень можна стверджувати, що використання методу візуального трансформера є ефективним для класифікації зображень та може бути використаним у подальших дослідженнях та розробках у цій області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR 2021*. P. 2-7

2. Wu B., Xu C., Dai X., Wan A., Zhang P., Yan Z., Tomizuka M., Gonzalez J., Keutzer K., Vajda P. Visual Transformers: Token-based Image Representation and Processing for Computer Vision. P. 1-13.

3. Google vit-base-patch16-224, URL: https://huggingface.co/google/vit-base-patch16-224 (дата звернення: 22.05.2023)

4. Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B. Swin. Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). 2021. P. 9992-10002. DOI: doi.org/10.1109/ICCV48922.2021.00986

5. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25 (NIPS 2012). P. 1-8

6. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015. P. 7-9

7. C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition CVPR, Boston, MA, USA, 2015, P 1-9, DOI: doi.org/10.1109/CVPR.2015.7298594.

8. He K., Zhang X., Ren S., Sun J. Identity Mappings in Deep Residual Networks. *ECCV*. 2016. P 630–645. DOI: https://doi.org/10.1007/978-3-319-46493-0_38

9. Tan M., Le, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *PMLR* . 2019. P. 6105-6114.

10. Chen X., Hu C.-J., Guo B. When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations. Computer Vision – *ICLR*. 2022 . P. 3-6. DOI: doi.org/arxiv-2106.01548

11. Sayak P., Pin-Yu C., Vision Transformers are Robust Learners. *AAAI*. 2021. P. 1-4. DOI: https://doi.org/10.1609/aaai.v36i2.20103

12. PythonRU, Алгоритм класифікації Random Forest на Python. URL: https://pythonru.com/uroki/sklearn-random-forest (дата звернення: 22.05.2023)

13. Google Developers. Gradient Boosted Decision Trees. URL: https://developers.google.com/machine-learning/decision-forests/intro-to-gbdt?hl=ru (дата звернення: 22.05.2023)

14. Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative Adversarial Nets. Advances in Neural Information Processing Systems. *NIPS* 2014.P. 139–144 DOI: https://doi.org/10.1145/3422622

15. Microsoft, resnet-152, URL: https://huggingface.co/microsoft/resnet-152 (дата звернення: 22.05.2023)

16. Timm, inception_v3.tf_in1k, URL: https://huggingface.co/timm/inception_v3.tf_in1k (дата звернення: 22.05.2023)

17. MarketsandMarkets. Artificial Intelligence - Mega Trends. URL: https://www.marketsandmarkets.com/mega_trends/artificial_intelligence (дата звернення: 22.05.2023)

18. Google Lens, URL: https://lens.google/ (дата звернення: 22.05.2023)

19. iNaturalist, URL: https://www.inaturalist.org/ (дата звернення: 22.05.2023)

20. Gurovich, Y., Hanani, Y., Bar, O. et al. Identifying facial phenotypes of genetic disorders using deep learning. *Nature Medicine*. Vol. 25. P. 60–64 .2019. DOI: https://doi.org/10.1038/s41591-018-0279-0

21. Number-OK, URL: https://number-ok.com/ru/ (дата звернення: 22.05.2023)

22. Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S. End-to-End Object Detection with Transformers. *ECCV*. 2020. P. 213–229. DOI: https://doi.org/10.1007/978-3-030-58452-8_13

23. Li Y., Mao H., Girshick R., He K. Exploring Plain Vision Transformer Backbones for Object Detection. *ECCV*. 2022.P. 280–296 DOI: https://doi.org/10.1007/978-3-031-20077-9_17

24. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser Ł., Polosukhin I. Attention is All you Need. *Advances in Neural Information Processing Systems*. 2017. P. 1-11

25. Yuan L., Chen Y., Wang T., Yu W, Shi Y, Jiang Z, Tay F., Feng J., Yan S. Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet. *ICCV*. 2021. P.1-10. DOI: doi.org/10.1109/ICCV48922.2021.00060

26. Chen C., Fan Q., Panda R. CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification. *ICCV*. 2021. P.1-10 DOI: doi.org/10.1109/ICCV48922.2021.00041

27. Lin T., Dollar P., Girshick R., He K., Hariharan B, Belongie S. Feature pyramid networks for object detection. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR*). 2017. P. 936-944. DOI: doi.org/10.1109/CVPR.2017.106

28. Ronneberger O., Fischer P., Brox T. Unet: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*. 2015. P. 234–241. DOI: https://doi.org/10.1007/978-3-319-24574-4_28

ДОДАТКИ

Додаток А. Приклади зображень для тестування програми

Клас «aircraft_carrier»

Клас «balloon»

Клас «castle»

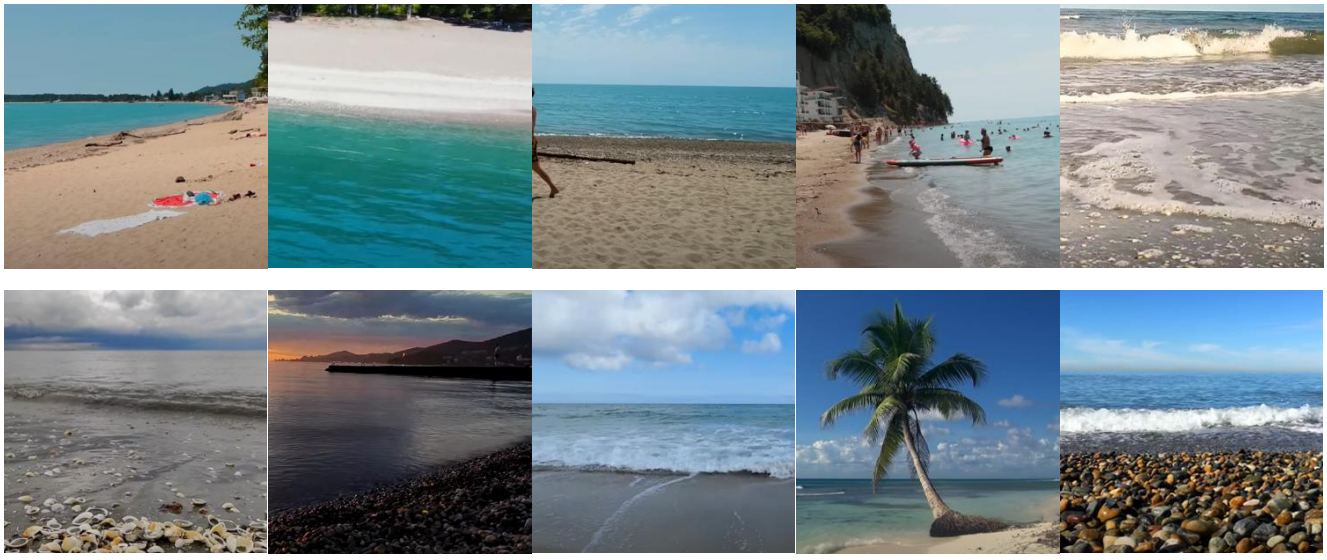
Клас «crayfish»

Клас «mantis»

Клас «pizza»

Клас «warplane»

Клас «seashore»



Клас «shoe_shop»

Клас «Pembroke»

Додатковий клас «sphinx_cat»

Додатковий клас «mouse_animal»